VIC rapport

ÅRGÅNG 3 NR 3
PRIS 15:- inkl moms



Hur gör man chips?
Planera din ekonomi
Assemblerskolan
Maskinspråksladdare för VIC 20

473-3

Tips och förslag



VIC rapport startar en bank i vilken vi samlar tips och förslag av alla sorter till din **VIC 20** och **VIC 64.** De kan innefatta allt ifrån sladdanslutning till maskinkods-programmering.

VIC rapport kommer att publicera dessa tips och förslag löpande.

Skriv ner dina tips och förslag och skicka dem till

VIC rapport, Tips och förslag
Box 420 54
126 12 Stockholm

Allt som publiceras belönas!

Ledaren

Så är det dags för ytterligare ett nummer av VIC-rapport. Vi på redaktionen ber så mycket om ursäkt för den försening som skedde vid förra numret. Vår förhoppning är att det inte skall behöva ske igen.

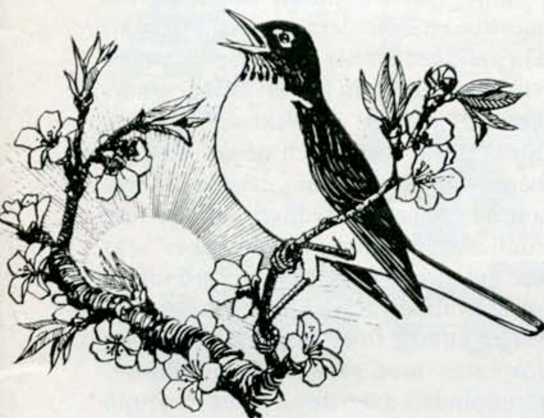
Som en del läsare kanske redan vet, har vi flyttat redaktionen från Göteborg till Stockholm. Samtidigt med flyttningen har vi lagt upp prenumerationsregistret på ett nytt datorprogram. Detta har inneburit en stor förändring av organisationen, vilket i sin tur påverkade utgivningen.

I dag är omorganisationen av redaktionen och uppdateringen av dataregistret klart. Redaktionen ser ljust på framtiden med många nya ideer i bagaget och någon försening ska inte behöva ske i framtiden.

Ett spännande område som behandlas i detta nummer är tillverkningen av chips. Jag var själv över på MOS Technology Inc i januari och fick lära mig hur denna tillverkning går till. Det var som att stiga in i framtiden. Människor var klädda som rymdmän (kliniskt rent) och teknologin samt precisionen var otrolig. Efter det besöket har jag full förståelse för om datorn krånglar vid tillfälle. Ja, jag har nästan vänt till att fascineras över att den *inte* krånglar oftare.

Hannover-mässan har, när detta läses, redan varit. Spännande reportage därifrån kommer att följa i nästa nummer.

Väl mött då och trevlig läsning tills dess.



Innehållsregister



VIC redaktion: Box 42054, 126 12 Stockholm
Ansvarig utgivare: Nina Linander
Administrativ redak.: Elisabeth Höglund
Teknisk redaktör: Matts Nilsson
Övriga medarbetare: Åke Hedman, Joakim Aspengren, Siv
Annonser: Telefon 08-744 59 20
Tryck: Dala-Offset AB— Falun
ISSN-nummer: 0281-8043

Annonsorder och annonsmaterial (helorginal eller negativ film) enligt överenskommelse under 1984.
Tryckförfarande: Offset.
Upplaga: 22.000 ex.

Nr 4 av VIC-rapport utkommer den 14 maj. Manusstopp för nr 5/6 är den 16 maj och annonsstopp den 18 maj.



Hej igen, äventyrare!

Hur har det gått för Dig? Fick Du ihop någon historia? Det enda Du måste tänka på när Du skriver är att inte göra det för svårt. Det ska ju gå att lösa också.

Allt i spelet måste ha sin förklaring. Oavsett om det rör sig om trolldom eller magiska krafter, så måste det framgå på ett eller annat sätt hur saker och ting skall vara eller göras.

Något som inte hör till det mest uppskattade i Adventurespel är långa sifferkombinationer, som spelaren måste komma på för att exempelvis öppna något. En sådan kan vara näst intill omöjlig att knäcka. Ord och fraser, som Du själv hittat på, och som skall sägas vid olika stadier i spelet, kan vara ännu svårare att komma på. Enklare, men också roligare blir det om det finns en logisk förklaring till siffer- eller bokstavskombinationerna lite här och var i spelet. Det bör också framgå i spelet när och hur spelaren ska göra något speciellt med ett föremål.

Ett bra tips för allt som är extra knepigt, är att lägga in en s k *hjälpfunktion* i spelet som talar om små nyttiga saker. Till exempel på svåråtkomliga platser eller vid speciella tillfällen. Risken med ett Äventyrsspel är att man "kör fast", därför är det bra med "snygg" vägledningshjälp.

Men, nu tänkte jag titta på vad Du gjort, samt visa ett utomordentligt enkelt sätt att ställa upp allting på. På så sätt har Du ett lätthanterligt manus när Du börjar programmera.

Vi börjar med det material som skall in i *datasatser*.

Dit hör:
— Vägval
— Rumsbeskrivningar
— Inventarier
— Övriga föremål i spelet, exempelvis dörrar och fönster, samt en del kodbeteckningar som datorn har till hjälp för att hålla ordning på allt Du skriver.

Har Du en oexpanderad VIC 20 (d v s VIC 20 utan extra minnestillsats) ryms det max ett spel (Adventure), skrivet på cirka fyra sidor i ett linjerat A4-block. För ett avancerat Adventure-spel på VIC 20 behöver Du minst 16 kb.

Om Du nu har fler än tio (10) rum eller saker, så måste Du ha en tilläggsrad i programmet som lyder
10 DIM AS(X), BS(X), CS(X)

'X' står för antal rum eller saker. Valet av variabelnamn, d v s 'AS, BS, CS' är naturligtvis upp till var och en. Men, använd inte datorns egna variabelnamn, som t ex TIS, eller TI (Du finner dem i manualen). I de här variablerna skall Du senare placera in det som skall in i *datasatserna.*

För att hålla ordning på alla variabler kan t ex namn som 'RUMS' och 'SAKS' med fördel användas. Tro nu bara inte att alla rum och saker skall ges ett egennamn. Nej, låt mig förklara.

D*IM-satsen* har till syfte att specificera en matrisvariabels storlek, samt att förbeställa plats åt denna i minnet. Det får finnas max 10 indexvariabler förutom DIM-satsen i datorns minne. Det betyder att alla rum får samma namn, men olika nummer. Mer om det där längre fram. Nu åter till listorna och vägvalsdelen.

I alla typer av sådana här spel är det enklast att se det hela som från ovan och därifrån gå i väderstreckens riktningar; *nord, syd, väst och öst.* För att göra det hela ännu enklare så gör vi följande. För att inte behöva skriva hela meningen "Gå nord", så skriver vi bara *'N'.*

Samtidigt med vägvalsdelen, tar vi också biten för när datorn håller redan på var någonstans Du är och vart Du tar vägen om Du nu går 'N'. Om allt stämmer skall Du hamna i ett annat rum, och då måste datorn veta vilket. Ta fram förra delen och titta på den karta jag ritade där, så ska jag förklara.

Titta på rum 5. Där ser Du att Du kan gå nord och öst. Om Du nu går 'N' i rum 5, så hamnar Du i rum 4, o s v. Har vi lite tur nu, så kommer vi till de här listorna. Så här ser det ut, inskrivet i en datasats.
DATA N4Ö6

(Har du fler rum än nio, skriver Du en nolla före varje siffra här upp till nio.)

Eftersom vi är i rum 5, där detta gäller, så fortsätter vi att skriva in *rumsbeskrivningen* på samma rad. Skilj rumsbeskrivningen från det tidigare med ett kommatecken (,).
DATA N4Ö6, RUM 5, NORD OST, o s v.
Samt vägvalspresentationen.

Skriv dessa listor så här.

(A)	(AS)	(BS)	(CS)
1	(vägvalskod)	(rumsbeskrivning)	(vägvalspresentation)
2	(vägvalskod)	(rumsbeskrivning)	(vägvalspresentation)
3	(vägvalskod)	(rumsbeskrivning)	(vägvalspresentation)
o s v.			

(A) = rumsnummer (skall inte skrivas in i datasatserna)
(AS) = datorns rumkontroll
(BS) = endast vad rummet eller platsen består av, t ex 'I EN STOR SAL' eller 'PÅ EN KYRKOGÅRD'. Alla inventarier och saker kommer in andra datasatser.
(CS) = Detta kommer upp på skärmen under spelets gång, för att inte tala om för spelaren vart han kan gå.

För att spara minnesutrymme ska Du inte skriva in 'JAG SER' före sakerna. De här orden behöver nämligen bara skrivas en gång, på utskriftsraden, senare i programmet.

Tips! Om Du vill att det skall vara mörkt i ett eller flera rum, tills dess att Du antingen tänder lampan eller har en viss egenskap som gör att Du ser i mörker, gör så här: Anteckna dessa rum först, alltså från 1 och nedåt. Det gör både hanteringen av data och programmeringen betydligt enklare om rum med samma "egenskaper" är samlade i grupper. Nummerordningen inom gruppen är oviktig, bara Du skriver in rum 1 först. Samma sak gäller för listan med saker och inventarier. Gruppindela även dessa. T ex en grupp för saker som man kan se, men inte ta på p g a deras magiska kraft.

Nu till sakerna: inventarier, dörrar, skatter m m.

(B)	(DS)	(PLATS)
1	En stor kista	1
2	En öppen kista	0
3	En låst dörr	2
4	En öppen dörr	0
o s v.		

(B) = Saknummer (skall inte skrivas in i datasatserna)

(DS) = Den för tillfället synliga beskrivningen av saken. Observera att det inte finns två kistor eller två dörrar, utan innan dörren öppnas är den en 'LÅST DÖRR' och efter att den öppnats en 'ÖPPEN DÖRR'. Båda beskrivningarna måste finnas med. Samma sak gäller för alla saker som kan ändras i spelet.

(PLATS) = Den här variabeln talar om för datorn i vilket rum Du önskar ställa exempelvis den här kistan. Observera att sakernas "utgångslägen" (hur de ser ut innan de ändrats av spelaren) har här siffran noll (0). Öppnar vi den här kistan, ändrar datorn bara (PLATS)-innehåll i rad 2 till 1, samt innehållet i rad 1 till 0. Därefter står det en 'ÖPPEN KISTA'. Hur datorn låter dessa byta innehåll, kommer jag till senare.

Det spelar ingen roll i vilken ordning Du skriver inventarierna, bara Du skriver in rätt rumsnummer på dem under (PLATS). Flera saker kan ha samma nummer under (PLATS).

Nu skriver Du in allt det här i datorn på följande sätt. OBS! Om Du har *både färre rum och saker än 10 så hoppar Du över alla DIM-satser.*

10 DIM AS(X), BS(X), CS(X)
20 FOR I=1 TO X: READ AS(I), BS(I), CS(I): NEXT
X = antal rum

Därefter skriver Du in datasatserna.

Observera att de här tre variablerna hör till ett och samma rum. Det innebär att om Du avslutar en datarad med annat än (CS), så måste Du börja nästa datarad med närmast efterföljande variabel. Att det inte står några radnummer här beror på att Du kan placera datsatser var som helst i programmet. Jag brukar lägga dem sist i programmen, så jag föreslår att Du börjar på rad 10000.

DATA N4ÖG, RUM 5, NORD OST, N255, RUM 4, NORD SYD, o s v.

Inläsningen av inventarierna ser ut så här.

30 DIM DS (X), PLATS (X)
40 FOR I=1 TO X: READ DS(I), PLATS (I): NEXT

X = antal saker.

Därefter är lätt att kolla om Du har gjort rätt. Har Du skrivit in inläsningsraderna och datasatserna samt kört 'RUN', så kollar Du enligt följande.

Skriv precis så här, men använd de variabelnamn Du skrivit.

FOR I=1 TO X: PRINT BS(I):NEXT
och tryck 'return'.

På skärmen skall Du nu kunna läsa alla rumsbeskrivningar. Om Du inte kan det, eller bara några stycken av dem i början, så har Du gjort fel. Kolla om Du kan ha glömt någon av de tre sakerna för varje rum. När Du hittat felet — rätta och skriv sedan kontrollraden igen, ända tills det bara finns rumsbeskrivningar på skärmen.

Inventarierna behöver också kontrolleras.

FOR I = 1 TO x: PRINT DS(I): NEXT

Nu skall bara inventarierna komma upp, inga siffror alltså. Om allt stämmer nu, så ligger samtliga uppgifter om till exempel rum 1 i AS(1), BS(1), CS(1), DS(1) och PLATS(1).

Jag skrev tidigare om *matris* och *indexvariabler.* Du har nu fem stycken *matrisvariabler,* var och en innehåller X stycken *indexvariabler.* Enkelt, eller hur?

B.T.



Glosförhör på VIC 64

Från Birger Gran fick vi följande program för glosförhör till Commodore 64

Här ett litet program till CBM 64:an som klarar av att lagra och förhöra valfritt antal (upp till 4 000) glosor. Man kan alltså spara sina glosor på kasettband mellan varje omgång, och fortsätta att mata in nya nästa gång. På så sätt slipper man besväret att mata in glosorna varje gång man vill förhöra sig själv. Ja, man kan kanske spara en hel termins (eller språkkurs) glosor, och ta fram efteråt för repetition.

Här en beskrivning av programmet:

Rad 5: Dimensionering och definition av en CR-sträng för datafilen. (CR = carriage return = vagnretur.)

Rad 10—50: Utskrift av meny, och nummerval, samt anrop av olika programrutiner vid nummerval från menyn.

Rad 60—120: Inmatning av svenska resp. engelska glosor. Efter varje inmatning glospar utskrift för kontrollläsning och möjlighet att göra om inmatningen av sista glosparet vid ev. fel. Inmatningen avslutas med tryck på Return när nästa svenska ord efterfrågas.

Rad 130—148: Listning av alla inmatade glas-par (10 i taget) för genomläsning och repetition före förhör.

Rad 150—250: Förhör svenska/engelska. Vid felsvar 1:a gång utskrift av första bokstav samt punkter för resterande bokstäver. 2:a gången fel fås 2 första bokstäverna och resterande antal som punkter. 3:e gången fel skrivs hela det rätta ordet (meningen) ut.

Rad 220: 'F' håller räkningen på antalet fel.

Rad 260: Subrutin som skriver ut rätt antal punkter enl. ovan.

Rad 300—380: Motsvarande rutiner som ovan men i motsatt riktning, d v s engelska till svenska.

Anm. Givetvis kan man ändra 'engelska' till valfritt annat språk.

Rad 400—430: Avslutning av förhöret, med utskrift av antal fel m m.

Rad 500—595: Rutin för att spara resp. hämta inmatade glosor till/från kasettband.

Hälsningar

Birger Gran

```
4 REM * GLOS-FöRHöR * AV BIRGER GRAN
5 DIMSV$(4000),EN$(4000):N=0:CR$=CHR$(13)
10 PRINT"* M E N Y *":PRINT"1.INMATNING AV GLOSOR":F=0
15 PRINT"2.LISTA INMATADE GLOSOR"
20 PRINT"3.FöRHöR SVENSKA/ENGELSKA":PRINT"4.FöRHöR ENGELSKA/SVENSKA"
30 PRINT"5.TILL/FRÅN DATAFIL":PRINT"ANTAL INMATADE GLOSOR: ";N
40 PRINT"NUMMERVAL";:NR$="":INPUTNR$:NR=VAL(NR$):IFNR=0ORNR>5THEN40
50 ONNRGOTO60,130,150,300,500
60 PRINT"* INMATNING AV GLOSOR *"
70 N=N+1:PRINT"NR ";N:INPUT"SVENSKA";SV$(N):IFSV$(N)=""THEN110
75 INPUT"ENGELSKA";EN$(N):PRINT"NR ";N;": ";SV$(N);" - ";EN$(N)
80 PRINT"KORREKT INMATAT";:J$="":INPUT"(J=JA)";J$
90 IFJ$<>"J"THENN=N-1:PRINT"MATA IN SISTA GLOS-PARET IGEN!"
100 GOTO70
110 N=N-1:J$="":PRINT"SLUT GLOSOR (J=JA)":INPUTJ$:IFJ$="J"THEN10
120 GOTO70
130 S=0:FORI=1TON:IFS=0THENPRINT"* INMATADE GLOSOR *"
140 PRINTI;":";SV$(I);" - ";EN$(I)
145 S=S+1:IFS=10THENS=0:INPUT"TRYCK <RETURN>";RE$
148 NEXT:INPUT "ÅTER TILL MENY  TRYCK <RETURN>";RE$:GOTO10
150 PRINT"* FöRHöR SVENSKA/ENGELSKA *"
160 FORI=1TON:PRINT"VAD HETER PA ENGELSKA: ";SV$(I):INPUTA$
170 IFA$=EN$(I)THENPRINT"KORREKT SVAR - BRA!":GOTO250
180 PRINT"LEDSEN - FEL! DET BöRJAR PÅ ";LEFT$(EN$(I),1);".";:GOSUB260
190 PRINT"FöRSöK IGEN!":INPUTA$:IFA$=EN$(I)THENPRINT"KORREKT.":GOTO250
200 PRINT"FEL IGEN! DET BöRJAR PÅ ";LEFT$(EN$(I),2);:GOSUB260
210 PRINT"FöRSöK IGEN!":INPUTA$:IFA$=EN$(I)THENPRINT"KORREKT.":GOTO250
220 PRINT"SVARET ÄR: ";EN$(I):F=F+1
250 NEXT:PRINT"TRYCK <RETURN>";:INPUTRE$:GOTO400
260 FORL=3TO(LEN(EN$(I))):PRINT".";:NEXT:PRINT:RETURN
300 PRINT"* FöRHöR ENGELSKA/SVENSKA *"
310 FORI=1TON:PRINT"VAD HETER PA SVENSKA: ";EN$(I):INPUTA$
320 IFA$=SV$(I)THENPRINT"KORREKT SVAR - BRA!":GOTO370
330 PRINT"LEDSEN - FEL! DET BöRJAR PÅ ";LEFT$(SV$(I),1);".";:GOSUB380
340 PRINT"FöRSöK IGEN!":INPUTA$:IFA$=SV$(I)THENPRINT"KORREKT.":GOTO370
350 PRINT"FEL IGEN! DET BöRJAR PÅ ";LEFT$(SV$(I),2);:GOSUB380
360 PRINT"SVARET ÄR: ";SV$(I):F=F+1
370 NEXT:PRINT"TRYCK <RETURN>";:INPUTRE$:GOTO400
380 FORL=3TO(LEN(SV$(I))):PRINT".";:NEXT:PRINT:RETURN
400 PRINT"AV ";N;" GLOSOR, HADE DU ";F;" FEL."
410 IFF=0THENPRINT"* B R A V O ! *"
420 IFF>2THENPRINT"DU BORDE STUDERA GLOSORNA MERA!"
430 PRINT"ÅTER TILL MENY, TRYCK <RETURN>";:INPUTRE$:GOTO10
500 PRINT"DATAFIL TILL/FRÅN BAND":PRINT"BAND I RÄTT POSITION (J=JA)"
510 J$="":INPUTJ$:IFJ$<>"J"THEN500
520 PRINT"VILL DU LAGRA GLOSOR (L)":PRINT"ELLER HÄMTA GLOSOR (H)"
530 INPUTJ$:IFJ$="L"THENGOSUB560
540 IFJ$="H"THENGOSUB580
550 GOTO10
560 OPEN1,1,1:PRINT#1,N
570 FORI=1TON:PRINT#1,SV$(I);CR$;EN$(I);CR$;:NEXT:CLOSE1:RETURN
580 OPEN1,1,0:INPUT#1,N
590 FORI=1TON:INPUT#1,SV$(I)
595 INPUT#1,EN$(I):NEXT:CLOSE1:RETURN

READY.
```

Användarvänlighet

Din dator kan vara lätt att använda. Men du kan på flera sätt förbättra detta när du gör dina program.



Din dator kan vara så användarvänlig på flera sätt. Den kan t ex som Vic ha ett stort och lättarbetat tangentbord och ge stora och lättlästa tecken på skärmen. Men utöver dessa goda förutsättningar har du själv en stor roll att göra dina program sådana att datorn framstår som vänlig mot användaren.

Det fanns t ex en gång en FORTRAN-kompilator som skrev ut SUCCESS! när man lyckades, det kändes härligt. Men vänlighet ska inte begränsas till gester, användaren mår bra av lite hjälp också.

Man kan se mer eller mindre akuta behov av den omtanken hos programmeraren inom tre områden:
In- och utdata hanteringen, d v s meny och fråge/svarshantering
Struktur, läsbarhet och kommentering i programmet
Rimlighetskontroll av indata, utdata och vissa interna beräkningar

När det gäller indata skall du låta datorn göra jobbet att översätta vanligt språk, och den skall också presentera vilka svar den accepterar. Här finns alla varianter av klurigheter i bruk, men jag tror man bör leva efter valspråket 'too clever is dumb'. Om du kräver att få strängarna 'ja' eller 'nej' och repeterar frågan om du får annat skräp är du ganska säker på att den som kör programmet vet vad han vill ha gjort. Ja, jag vet att man lätt blir otålig på en dum dator, man vill skriva 'j' eller 'n' eller helst bara nicka eller skaka på huvudet. Jag tro ändå att konversationen mår bra av fasta regler. Men vad är det man ska svara på? Det är vänligt och omtänksamt att ha definierat ofarliga siffervärden i förväg i programmet, skriva ut dem och fråga om de passar bra. Om inte, går man till en inputrutin där tillåtet talområde står i frågetexten och en test görs på inmatat värde.

Ovänligt? Ja, om du vill kunna lita på datorn så får du låta den vara lite misstrogen mot dig själv.

Du har väl hört om GIGI-principen som gäller för all databehandling: 'GARBAGE IN = GARBAGE OUT' Erfaret folk brukar t o m säga 'GOSPEL OUT'.
(garbage = skräp, gospel = tro, evangelium)

Tro inte nu att det är stenhårt omöjligt att vara lite smidig på den här punkten. I system som börjar närma sig vad man kallar Artificiell Intelligens kan användaren ange vilken nivå av kompetens han anser sig ha och få tillbaka sin självkänsla genom att han får lov att använda egna förkortningar och att programmen inte ställer så mycket frågor.

Att han dessutom är övervakad och hans prestationer i antal systemkvaddförsök etc. egentligen styr kompetensklassningen behöver han ju inte veta.

Åter till vad du bör tänka på när du programmerar.

Efter input av data kan det hända att resultatet dröjer. Då bör man av ett modernt begåvat program få kvittensen 'vänta lite' eller 'ta en paus medan jag räknar'. Det är mycket trevligare än att bli sittande framför en tom skärm och undra vad som hänt. Om du gör ett spel så är presentationen givetvis en del av spelets ide och man ska inte diskuter den, men en trevlig vinkel på ett spel är att det tar in namnet på den som spelar och kommer ihåg hans resultat till ett protokoll.

I mera traditionella databehandlingsuppgifter är det viktigt att den som ska ta hand om resultatet får det i den form han eller hon behöver och att alla förutsättningar och sidodata också redovisas.

När du som användare av t ex ett Basic-program finner att du skulle vilja ändra lite på det så kommer du i kontakt med den inre användarvänligheten. Det är bara om programmet är uppdelat i lättförståeliga bitar, och har förklaringar till variabelnamnen som du har en god chans att lyckas rota i ett program som någon annan gjort från början. Lägg därför själv ned omtanke på detta och tänk särskilt på två saker:
1) Skilj mellan väsentliga, permanenta variabelnamn och s k slaskvariabler som du använder i oberoende små beräkningar av mellanresultat.
2) Lämna aldrig kvar kommentarer från en gammal programversion utan att kontrollera att de fortfarande är meningsfulla och riktiga. Det är så frestande att tro på kommentarerna men datorn struntar som bekant i dem. Vi kanske får AI i framtiden som gör självkommenterande program, men än så länge är den bästa vägen att välja variabelnamn så förklarande som möjligt och skriva kommentarer inte om vad man gör utan om varför!

Rimlighetskontroll slutligen har vi redan nämnt i input-sammanhang. Basic-tolken t ex är ju barsk men vänlig som stoppar felskrivna kommandon, och programmet kan bli vänligare = säkrare om du går igenom dina variabler, ser vilka talvärden de kan variera mellan och lägger in test som slår larm för orimliga värden. Och, väl värt att observera, det är ofta i sådana handgripliga, formella studier av programmen som logiska galenskaper uppdagas; gör det tidigt och bli en vänlig användare!

Läst sen sist . . .

Den maskin det händer mest med just nu är 64-an, och det ser man om inte annat när det gäller böcker. På Studentlitteratur har man översatt William Turner's LÄR DIG ANVÄNDA VIC-64, som för övrigt är en i en lång rad LÄR DIG-böcker (i utgivningsplanen finns också en för PET med).

Författaren börjar helt och hållet från början, och klarar av det mesta, inklusive ljud och animering på 100 sidor. Men Geije Johansson, som står för den svenska översättningen, kan knappast ha gjort mycket annat än översatt, för det är väl många fel i texten. Garafik med låg upplösning kan knappast påverka skärpan i bilden. Och vad händer om man sätter ett GOTO i en datasats? I CP/M och MSDOS stoppar CTRL/C en programkörning, men mig veterligt lämnar detta kommando 64-an helt oberörd.

Bland alla irriterade småfel finns det också rena korrekturfel. Vad är MUÅÅER? Och vad menas med INAMTADE? Det här är inte likt Studentlitteratur. De som köper böcker av den här typen är ovana vid datorer, och de har helt enkelt inte kunskapen som behövs för att veta vad som är rätt och vad som är fel. Bokköparna måste ha rätt att kräva exta noggrann faktagranskning och korrekturläsning när det gäller databöcker (säger han i glashuset). Och kommandot RUN används definitivt inte för att ladda ett program från kassett. Men samtidigt saknas snabbladdningskommandot SHIFT + RUN/STOP.

Bildmaterialet är utmärkt, även om jag personligen ifrågasätter värdet av en 64 med en floppy bredvid och en sladd i bakgrunden. Bildtexten lyder "Diskettenhet ansluten till VIC-64". Jag är också tveksam till bilderna som visar meddelanden på bildskärmen. Enligt dem, har 64-an 22 teckens bildskärm med 22 rader. Ett ganska onödigt sätt att förvirra läsaren, som definitivt inte kan känna igen sig.

Förhoppningsvis kommer all dessa småfel att rättas till i nästa upplaga, och då har det blivit en alldeles utmärkt liten bok för nybörjaren.

För den mer försigkomne

Liber har också givit sig in på databoks-marknaden. AVANCERAD PROGRAMMERING på VIC 20 OCH VIC 64 heter en bok av Arne Kullbjer och Christer Ohlman. Den är mycket ambitiöst upplagd, även om jag tillåter mig ha en egen definition av ordet 'avancerad'. Och enligt den definitionen finns det ingenting avancerat i den här boken. Men för den som klarat av alla nybörjarböcker och vill läsa mer, är det här ett förträffligt alternativ.

Vad som sagts ovan om korrekturläsning gäller även för AVANCERAD PROGRAMMERING. Varför skall det vara så omöjligt att producera en databok med alla programlistor helt korrekta? Är det sådant sug i marknaden att konsumenterna sväljer vad som helst? Med den här boken är det kanske inte så farligt, eftersom läsaren enligt titeln bör vara betydligt mer än nybörjare.

Boken är hur som helst rejält upplagd, med massor av exempel och övningsuppgifter (samt facit). Grafiskt har man också arbetat ordentligt med boken: så är till exempel alla program i fetstil med större tecken än den övriga texten. Detta uppskattas mycket av en som är mer van vid förminskade programlistor från matrisskrivare med dåligt färgband. Köp den här boken, men tro inte att du blir proffs bara för att du läst den. Men det är klart, en bit på väg kommer du alltid.

J. Stiernborg

Append — Merge

Stefan Holmqvist, 13 år, skrev till oss med ett manus, som vi publicerar:

Append—Merge

BASIC-kommandot "Append" eller "Merge" används till att lägga till ett program till ett i minnet redan befintligt program utan att man behöver skriva om det. Men det fordras att alla radnummer i det andra programmet är högre än i det första. Detta kommando, som inte finns i VIC-20 kan lätt stimuleras med BASIC. Det kan genomföras med följande kommando i direktmod enligt följande:

Ladda in det första programmet.

Anteckna följande värden vid kommando PRINT PEEK(43);PEEK(44). Dessa minnespositioner inneåller startadress för BASIC-programmet.

```
A = PEEK(45) + PEEK(46)*256-2:PRINT A
```

Minskningen med 2 utförs för att de sista stopptecknen i BASIC-strängen inte skall komma med vid sammanslagningen. De två adresspositionerna i minnet som pekar på var BASIC-programmet ska laddas, ändras nu före laddningen av det andra programmet med kommandot:

```
POKE 43,A-INT(A/256)*256:POKE 44,INT(A/256)
```

Nästa program skall laddas in med kommandot LOAD på vanligt sätt med eller utan programnamn.

Kommandorutinen LOAD använder ovanstående pekare och sköter laddningen automatiskt med start vid nämnda adress.

Då programladdningen är klar, har det andra programmet laddats in direkt efter det första utan avbrott. Men saken är inte helt klar ännu.

Innan programmet kan köras måste pekarna som anger startadressen för BASIC-programmet återställas. Detta sker genom att de tidigare antecknade värdena som pekare 43 och 44 innehöll, laddas in på nytt med t ex POKE 43,1:POKE 44,16.

Programmet kan nu editeras, köras och sparas som ett enhetligt program, ja man kan behandla det som ett program, som skall utökas och lägga till ytterligare program på program på det sätt som beskrivits.

Stefan Holmqvist

TIPS: Montera din dator

Upp och nermonteringen av anläggningen tar en del tid. Dessutom så sliter det på kontakterna. För att göra det lätt för oss så har vi monterat anläggningen på en spånskiva. Nu är det bara att ställa allt i ett hörn när datorn inte används.

Material du behöver

Om du skall montera upp din utrustning så behöver du följande:
1 vitlackerad spånskiva i lämpligt format.
2 meter plastat stålband.
Klammer för montering av kuloledning.
Stålband och klammer finns i elaffärer.
Plåtskruv, och brickor.
3 vinkeljärn.
1 handtag till kökslådor. 1 Banankontakt. 1 slangklämma.
1 polskruv som passar till banankontakter. Elledning.
4 filttassar som passar till stolsben.

Sätt igång och bygg

Montera vilkeljärn på handtaget. Skruva fast vinkeljärnen i framkanten på spånskivan. Om du använder plåtskruv så fäster det bra i spånskivan. Nu kan du bära skivan bekvämt.
Lägg bandstationen och datorn på skivan. Tänk på att anslutningen för el och joystick kräver utrymme på högersidan.

Bocka till bandet så det passar datorn. Bandet går lätt att klippa av med en avbitartång, då det är stansade hål i stålbandet. Bocka bandet så det spänner lagom mot datorn. Använd brickor så inte skruvarna åker genom bandet. Var noga med monteringen av bandet så det inte hindrar Commodoretangenten, eller manövreringen av räkneverkets nollställare. Om du har extraminne, spelartridge, eller utrustning till användarporten, så plugga in det. Nu ser du var du kan placera transformator och antennfilter (VIC-20). Elledningen till datorn virar du runt transformatorn innan du sätter fast den med stålband. Tänk på att du lätt skall kunna nå anslutningen för TV på antennfiltret när du sätter fast kabeln. Kabeln till bandstationen rullar du ihop i en ring och sätter fast den. För att inte kablarna skall haka i någonstans när du bär utrustningen, så sätter du fast dessa med klammer.

Med denna kompakta anläggning är det risk att datorn stör bandaren. Därför bör du jorda bandstationens kabel.

Skruva fast det tredje vinkeljärnet. Sätt fast bandarens skärmkabel med polskruven. Drag en kabel från ett vattenburet element, eller från ett vattenledningsrör. Lämplig kabel är RK 0.75 mm2.
Använd en slangklämma när du monterar kabeln i elementet, så det blir bra kontakt. I den andra änden monterar du en banankontakt. Till dig som bara har element hemma vill vi ge följande råd. Försök aldrig att jorda något i ett elektriskt föremål, såsom spis, kylskåp, eller element. Om det blir ett elfel så kommer det att gå en strömstöt in i din dator. Denna tusendels sekund är tillräcklig för att VIC-chipset skall gå sönder.

Om någon plåtskruv har gått sönder genom spånskivan, så måste du klippa av den lite. Annars är det risk att du river dig.

När allt är färdigt så limmar du fast filttassarna. Nu är det ingen risk att spånskivan repar bordet.

Allt detta kostar cirka 150:— kronor. Det kan verka mycket. Men genom att minskar slitaget på datorn och gör att det är lättare att ta fram datorn så är det en bra investering.

Skäftingebackens datorörnar/
Alf Olsson



RTTY program för VIC-20 med expanderat minne

RTTY (Radio TeleType) har existerat inom amatörradiovärlden i många år, men endast med mekaniska maskiner. I och med datorns insteg i vardagen så har en ny möjlighet för RTTY instresserade öppnat sig. Här presenteras ett RTTY program för VIC-20 med minst 8K minnesexpansion.

VIC-datorerna är mycket lämpade för datakommunikation eftersom de har ett förberett RS 232 snitt. Där ställer man i två adresser BAUD hastigheten, antal bitar som varje tecken har, antal stopp bitar etc.

Vid datakommunikation sänds eller mottages data som en följd av pulser (+ 5V och jord). En byte sänds i åtta bitar samt start- och stoppbitar. Detta beskrivs ganska bra i VIC REVEALED sid 204—215. I mitt program, skrivet för VIC-20, behövs en minnesexpansion av minst 8 K. Naturligtvis går programmet att banta ner så att det passar en oexpanderad VIC. Det är också relativt enkelt att skriva om det för VIC-64.

Jag har i programmet gett möjlighet att sända och ta emot i både ASCII- och BAUDO kod som nog är den vanligaste sändningstypen över radio.

I BAUDO koden representeras varje tecken av 5 bitar som hämtas ur en tabell (se rad 10 000—10 100), och för att täcka in alla bokstäver och siffror används ett bokstav- och sifferskift. Detta sköts automatiskt i programmet. ASCII representationen är densamma som i datorn och detta medför lite enklare hantering av mottagning och sändning för varje tecken.

När man startar programmet får man välja mellan ASCII eller BAUDO samt möjligheten att skriva in dagens datum och rätt tid. Som sista delen i uppstarten får man ange den hastighet som datat skall överföras. Hastigheten ställs på raderna 280—314.

Under programmets gång finns det en del funktioner som kontrollerar sändning och mottagning.

Under mottagning:
PF2 Tillbaka till start menyn
PF3 Övergång till sändning

Under sändning:
PF1 Övergång till mottagning
PF2 Bild meny
PF4 Datum och tid sänds
PF5 DE SMØNAM SMØNAM SMØNAM sänds
PF6 RYRYRYRYRYRYRYRYRY-RYRY sänds
PF7 PSE KKK sänds
PF8 CQ CQ CQ CQ CQ CQ CQ CQ sänds
Inmatning av QTC (personligt meddelande)
Inmatning av motstationens signal
hme WRZ DE SMØNAM sänds (eller motstation)

Vid sändning läggs en signal ut på porten (PB7) som vid inkoppling kan dra PTT:n.

I programkoden ligger nu mina egna texter och anropssignal som du själv kan byta ut. T.ex på rad 140 i variabeln M3S ligger din egen anropssignal. Även i de olika sändningsbilderna går egen text att lägga in. Det system som texterna läses in efter kan du själv se i raderna 2140—2270. Programmet går att modifiera till var och ens egen personliga stil.

Inkopplingen till VIC från ett modem sker via user port enligt följande:
— Ut från dator in på AFSK stift M (CB2)
— In till dator från demodulator stift B och C (CB1 och PBO) som kopplas ihop.
— Gemensam jord på stift A (GND)
— PTT:n kopplas på stift L (PB7)

Tänk på att vid RESET av datorn eller vid körning av annat program så ligger + 5V ut på stift L (PB7). Sedan kan det behövas en invertering av signalerna på stift B, C eller M beroende på vilken Demodulator/AFSK man har.

PIN	TYPE	RS232	FUNCTION
A	GND	GND	Protective ground
B	CG1	Sin	Received data
C	PBO	Sin	Connected to Sin
D	PB1	RTS	Request To Send
E	PB2	DTR	Data Terminal Ready
F	PB3	RI	Ring Indicator
H	PB4	DCD	Received line signal
J	PB5		
K	PB6	CTS	Clear To Send
L	PB7	DRS	Data Set Ready
M	CB2	Sout	Transmitted Data
N	GND	GND	Signal ground

Inkopplingsbeskrivning på User Port

```
5 REM                                              *RADIO TELETYPE
6 REM                                              *AV DAG WINDARP
7 REM
10 PRINT"♥":CLOSE 2
11 POKE 36879,25:POKE 36878,15:POKE 37138,134:POKE 37136,127
20 PRINT"■■■◣   ■RADIOTELETYPE■■■■"
28 PRINT"■■■■B=BAUD   A=ASCII■■■■"
30 PRINT"■◣MODE      :      ■B■■■";:INPUT M2$
32 PRINT"■                        ■"
40 IF(M2$<>"A")AND(M2$<>"B")THEN 10
110 IFM2$="B"THEN OPEN2,2,3,CHR$(32+64)+CHR$(16):GOTO 130
120 IFM2$="A"THEN OPEN2,2,3,CHR$(0)+CHR$(16):GOTO 130
130 D1$="840000":M1$="QRZ    ":A1$="N"
140 M3$="SM0NAM":REM                               *EGEN ANROPSIGNAL
141 T2=23:REM                                      *ANTAL BILDTEXT RADER
170 REM
```

```
171 DIMAS%(62),AC%(62),T1$(100),T2$(20)
172 FORI=1TO62:READ AS%(I):NEXT
173 FORI=1TO62:READ AC%(I):NEXT
174 FORI=1TOT2:READ T1$(I):NEXT
180 PRINT"◣DATE      :■    "D1$"■■■■■■■■■";:INPUT D1$
190 PRINT"◙◣TIME      :■    "TI$"■■■■■■■■";:INPUT TI$
200 PRINT"◙◙◙45 50▊ 75 100 110 300▔▔▔▔"
240 PRINT"◙◣HASTIGHET:■    45■■■■";
242 INPUT SP
244 PRINT"◙                          ▔▔"
250 PRINT"◙◣AUTOIGNOR:■    "A1$"■■■";:INPUT A1$:A1$=LEFT$(A1$,1)
252 IF(A1$<>"J")AND(A1$<>"N")THEN250
280 IFSP=45THEN L=60:H=95:GOTO 400
290 IFSP=50THEN L=148:H=86:GOTO 400
300 IFSP=75THEN L=187:H=56:GOTO 400
310 IFSP=100THEN L=74:H=43:GOTO 400
312 IFSP=110THEN L=90:H=39:GOTO 400
314 IFSP=150THEN L=220:H=28:GOTO 400
320 GOTO 200
400 POKE665,L:POKE666,H
402 IFPEEK(659)=0 THEN GOTO 5000
410 FL=0
420 PRINT"◥◣RX:■"SP"◣BAUD■◙■■■■■■■■■■■■■"
430 POKE667,PEEK(668)
440 GET#2,BA$
450 IF BA$="" THEN 530
460 BA%=ASC(BA$)
470 IFBA%=27 THEN FL=1:GOTO 530
480 IFBA%=31THENFL=0:GOTO 530
490 IFFL=1THENPR%=AS%(BA%+31):GOTO 510
500 PR%=AS%(BA%)
510 PRINTCHR$(PR%);
530 GETA$:IFA$="■"THENGOTO 10
540 IF A$="■"THEN GOTO 420
550 IFA$="■"THENPOKE659,224:GOTO 580
560 IFA$="■"THENFL=00
570 GOTO 440
571 REM                                                *BAUDO
580 PRINT"◥◣TX:■"SP"◣BAUD■◙■■■■■■■■■■■■■":P=0:SF=0
600 POKE37138,255:POKE37136,129
610 GETB$:IFB$="■"THENGOSUB850:POKE659,96:POKE37138,134:POKE37136,127:GOTO420
611 IFPEEK(669)<>PEEK(670)THEN620
612 IFA1$="J"THEN A1=A1+1
615 IF(A1$="J")AND(A1=12)THENPRINT#2,CHR$(31);:A1=0:SF=00
616 REM                                                *VAL MENY FOR BAUDO
620 IFB$="■"THENGOTO2000:REM                           *PF2
630 IFB$="■"THENB$="":REM                              *PF3
640 IFB$="■"THENB$="DATE:"+D1$+" TIME:"+TI$:REM        *PF4
650 IFB$="■"THENB$="DE SM0NAM SM0NAM SM0NAM ":REM      *PF5
660 IFB$="■"THENB$="RYRYRYRYRYRYRYRYRYRYRYR":REM       *PF6
670 IFB$="■"THENB$=" PSE K K K ":REM                   *PF7
671 IFB$="■"THENB$="CQ CQ CQ CQ CQ CQ CQ ":REM         *PF8
672 IFB$="↑"THENGOSUB3000:REM                          *QTC INPUT
674 IFB$="←"THENGOSUB4000:REM                          *MOTSTATION INPUT
675 IFB$="■"THENB$=M1$+" DE "+M3$
680 IFB$=""THEN610
690 PRINT B$;
720 GOSUB1000
730 GOTO610
850 IF PEEK(669)=PEEK(670)THEN POKE 36876,200:FORX=1TO300:NEXT:POKE36876,0:RETU
RN
860 GOTO 850
1000 REM                                               *SANDNING BAUDO
1010 IFB$=CHR$(13)THENPRINT#2,CHR$(8);:PRINT#2,CHR$(8);:PRINT#2,CHR$(2);:GOTO10
40
1012 IFASC(B$)<32ORASC(B$)>93 THEN 1150
1020 GOTO1050
1040 PRINT#2,CHR$(31);:PRINT#2,CHR$(31);:P=00:SF=00:GOTO1150
1050 IFLEN(B$)>1THEN P=P+LEN(B$)
1060 P=P+1:IFP>=58ANDB$=" "THENP=0:B$=CHR$(13):GOTO 1010
1070 IF P=>65 THEN B$=CHR$(13):GOTO1010
1080 FORI=1TOLEN(B$):UT$=MID$(B$,I,1)
1090 U%=ASC(UT$)-31
```

```
1100 UT%=AC%(U%)
1110 IFUT%<0ANDSF=00THENPRINT#2,CHR$(27);:SF=01
1120 IFUT%>=0ANDSF=01THENPRINT#2,CHR$(31);:SF=00
1130 PRINT#2,CHR$(ABS(UT%));
1140 NEXT
1150 RETURN
1200 REM                                          *STRANG TX
1220 FORI=1TOLEN(B$):UT$=MID$(B$,I,1)
1230 U%=ASC(UT$)-31
1240 UT%=AC%(U%)
1250 IFUT%<0ANDSF=0THENPRINT#2,CHR$(27);:SF=01
1260 IFUT%>=0ANDSF=01THENPRINT#2,CHR$(31);:SF=00
1270 PRINT#2,CHR$(ABS(UT%));
1280 NEXT
1290 PRINT#2,CHR$(8);:PRINT#2,CHR$(8);:PRINT#2,CHR$(2);
1300 P=00
1310 RETURN
2000 REM                                          *MENY FOR BILDSANDNING
2010 PRINT"■■":P=0:SF=0
2020 PRINT"■BILDER..."
2030 PRINT"■■UTRUSTNING  ■(1)"
2040 PRINT"■             ■(2)"
2050 PRINT"■QTC...       ■(3)"
2060 PRINT"■             ■(4)"
2065 PRINT"■QRZ QRZ      ■(5)"
2070 PRINT"■             ■(6)"
2080 PRINT"■             ■(7)"
2090 PRINT"■             ■(8)"
2100 PRINT"■73 DE...     ■(9)"
2110 PRINT"■■ANGE ALT.NR(1-9)■"
2120 GETA$:IFA$=""THEN2120
2130 IF(A$>"9")OR(A$<"1")THEN580
2140 A=VAL(A$):P=0:REM                           *SELEKTERING AV RATT BILD
2141 PRINT#2,CHR$(8);:PRINT#2,CHR$(8);:PRINT#2,CHR$(2);
2150 IFA=1THENFORX=1TO9:B$=T1$(X):PRINTB$:GOSUB1200:NEXTX
2165 IF(A=3)AND(T2$(1)="")THEN610
2170 IFA=3THENFORX=1TO8
2172 IFA=3THENIFT2$(X)=""THEN NEXTX:GOTO610
2173 IFA=3THENB$=T2$(X):PRINTB$:GOSUB1200:NEXTX
2175 IFA=3THENPRINT#2,CHR$(8);:PRINT#2,CHR$(8);:PRINT#2,CHR$(2);:PRINT#2,CHR$(2
)
2176 IFA=3THENPRINT#2,CHR$(2);:PRINT#2,CHR$(2);:PRINT#2,CHR$(2);
2178 IFA=5THENB$=T1$(23):PRINTB$:GOSUB1200
2270 IFA=9THENFORX=10TO22:B$=T1$(X):PRINTB$:GOSUB1200:NEXTX
2400 GOTO610
3000 REM                                          *QTC DATA
3010 PRINT"■":REM                                 *INMATNING AV QTC
3020 FORI=1TO8:T2$(I)="":NEXT
3030 PRINT"■QTC TILL...■■■"
3040 I=1
3042 INPUT"■CALL  :■";A$
3043 IFA$=CHR$(13)THEN3050
3044 T2$(1)=A$+" "+A$
3050 PRINT"■RAD■"I"■:■";
3060 INPUT T2$(I+1)
3070 IFT2$(I+1)=""THEN3090
3075 IFI>=7THEN3090
3080 PRINT:I=I+1:GOTO3050
3090 PRINT"■■QTC TILL...■"
3100 FORI=1TO8
3110 PRINTT2$(I)
3120 NEXTI
3130 PRINT"■OK ? (J/N)■"
3140 GETX$:IFX$=""THEN3140
3150 IFX$="N"THEN3030
3160 PRINT"■":GOTO580
4000 REM                                          *MOTSTATION
4010 PRINT"■■"
4020 PRINT"■MOTSTATION    ■"M1$"■■■■■■■■■";
4030 INPUT M1$
4040 IFLEFT$(M1$,3)="QRZ"THENM1$="QRZ    "
4050 PRINT"■":B$=""
4060 RETURN
```

```
5000 REM
5010 REM                                                    *ASCII SANDNING
5020 REM
5030 REM
5040 PRINT"▖RX:■"SP"▖BAUD■▖■■■■■■■■■■■■■"
5050 POKE667,PEEK(668)
5060 GET#2,BA$
5070 IF BA$="" THEN 5140
5130 PRINTBA$;
5140 GETA$:IFA$="■"THENGOTO10:REM.PF1 START
5160 IFA$="■"THENPOKE659,224:GOTO6000:REM.PF3 TX ASCII
5180 GOTO5060
6000 PRINT"▖TX:■"SP"▖BAUD■▖■■■■■■■■■■■■■":P=0
6010 POKE37138,255:POKE37136,129
6020 GETB$:IFB$="■"THENGOSUB850:POKE659,0:POKE37138,134:POKE37136,127:GOTO5000
6030 IFPEEK(669)<>PEEK(670)THEN6060
6060 REM                                                    *VAL MENY FÖR ASCII
6070 IFB$="■"THENB$="CQ CQ CQ CQ CQ":REM                    *PF3
6080 IFB$="■"THENB$="DATE:"+D1$+" TIME:"+TI$:REM            *PF4
6090 IFB$="■"THENB$=" DE SM0NAM SM0NAM ":REM                *PF5
6100 IFB$="■"THENB$="RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYR":REM*PF6
6110 IFB$="■"THENB$=" PSE K K K ":REM                       *PF7
6130 IFB$=""THEN6020
6140 PRINTB$;
6150 FORI=1TOLEN(B$):UT$=MID$(B$,I,1):REM                   *SÄNDNING AV ASCII
6160 PRINT#2,ASC(UT$);
6170 NEXTI
6180 GOTO6020
10000 REM                                                   *TECKEN DATA
10010 DATA 69,10,65,32,83,73,85,13,68,82,74
10020 DATA 78,70,67,75,84,90,76,87,72,89,80
10030 DATA 81,79,66,71,00,77,88,86,00
10040 DATA 51,10,45,32,39,56,55,13,113,52,7
10050 DATA 44,93,58,40,53,43,41,50,92,54,48
10060 DATA 49,57,63,91,00,46,47,61,00
10070 DATA4,-11,-5,31,31,31,31,-5,-15,-18,31,-17,-12,-3,-28,-29,-22
10080 DATA-23,-19,-1,-10,-16,-21,-7,-6,-24,-14,-14,31,-30,31,-25,31
10090 DATA3,25,14,9,1,13,26,20,6,11,15,18,28,12,24,22,23,10,5,16,7
10100 DATA30,19,29,21,17,-26,-20,-13
10120 REM                                                   *DATA FÖR MENY BILDER
10125 DATA"----------------------------------":REM         *INDEX 1
10130 DATA"UTRUSTNING.."
10140 DATA"2M - IC 251E 1-50W OUTPUT"
10150 DATA"10-80M - IC 740"
10160 DATA"10 EL KRYSS YAGI AND 10-20M VERTICAL"
10170 DATA"DATOR VIC 20 MED EGEN PGM VARA"
10180 DATA"BAUDO OCH ASCII 45-300 B"
10190 DATA"MM MM ... "
10200 DATA"----------------------------------":REM         *INDEX 9
10220 DATA"  ---------------":REMT                         *INDEX 10
10230 DATA"                  "
10240 DATA"   77777   333   "
10250 DATA"       7  3   3  "
10260 DATA"      7       3  "
10270 DATA"     7      33   "
10280 DATA"    7         3  "
10290 DATA"    7     3   3  "
10300 DATA"    7      333   "
10305 DATA"                  "
10310 DATA"  ---------------"
10320 DATA"                  "
10330 DATA"    DE SM0NAM":REM                                *INDEX 22
10350 DATA"QRZ DE SM0NAM ":REM                               *INDEX 23
63000 END
READY.
```

Ett program som gör ett program!

Ofta vill man från ett BASIC-program ladda en minnesarea. Det kan gälla ett maskinkodsprogram, egendefinierade tecken eller något annat. Enklast och snabbast gör man detta genom att i DATA-satser definiera minnesinnehållet och med en FOR-READ-POKE-NEXT-loop mata ut minnesinnehållet. Att bestämma varje individuellt data i decimal form är däremot ofta tidsödande och tråkigt. (Kodningen skall vi bara inte tala om.) Inmatningen av maskinkodsprogram eller tecken är kanske redan gjord från en maskinkodsmonitor eller i hexa-decimal form. I så fall kan det bifogade programmet vara till stor hjälp. Radnummer 100—990 är reserverade för REM-satser. Hur många REM-satser som behövs beror på hur stor datamängd som skall sparas. Det är inte fel att ha för många REM-satser. Observera att det är lätt att skapa kopior av t ex rad 100. Det är bara att gång på gång ändra radnumret och trycka på RETURN. Resten av programmet byter ut REM-satserna mot DATA-satser med rätt data-innehåll samt avslutar med en utmatningssats. Efter att detta program har körts så har man alltså ett helt nytt program i minnet! Glöm därför inte att SAVE:a programmet innan det körs! Efter körningen kan du tillfoga ett eget program ovanpå det konstruerade. Har du programmeringshjälp så kan du göra RENUMBER och MERGE. En liten avslutande varning dock. Programmet ställer inte om pekaren för BASIC-toppen d v s datorn tror att det skapade programmet upptar lika stort utrymme som det ursprungliga programmet. Normalt gör inte detta något. Du kan t ex köra programmet precis som vanligt. Spara och ladda programmet så är problemet löst.

Programmet är gjort för en VIC-20.

```
1 PRINT"♥":D=0
2 PRINT"INSTRUKTIONER?"
3 PRINT"TRYCK 9 INOM 5 SEK":C=0
4 FORN=1TO800:IFPEEK(197)=4THENC=1
5 NEXT
10 PRINT"♥"
11 T1=TI$/60:D=D+1
12 S=0:DX=0:DY=0
13 POKE36879,88
15 IFD=300THENFORN=7726TO8122STEP22:POKEN,102:NEXT:FORN=38446TO38842STEP22:POKEN
,7:NEXT
16 IF D=300THENFORN=7743TO8139STEP22:POKEN,102:NEXT:FORN=38463TO38859STEP22:POKE
N,7:NEXT
17 IFD=300THEND=0
20 POKE7701,81
30 POKE38400+7701-7680,2
40 POKE7930,81
45 POKE38400+7930-7680,6
50 U=21:V=0:X=08:Y=11
55 FORN=7680TO7723:POKEN,102:NEXT
56 FORN=38400TO38443:POKEN,7:NEXT
57 FORN=8142TO8186:POKEN,102:NEXT
58 FORN=38862TO 38905:POKEN,7:NEXT
59 FORN=7724TO8120STEP22:POKEN,102:NEXT
60 FORN=38444TO 38840STEP22:POKEN,7:NEXT
61 FORN=7725TO8121STEP22:POKEN,102:NEXT
62 FORN=38445TO 38841STEP22:POKEN,7:NEXT
63 FORN=7744TO8140STEP22:POKEN,102:NEXT
64 FORN=38464TO 38860STEP22:POKEN,7:NEXT
65 FORN=7745TO8141STEP22:POKEN,102:NEXT
66 FORN=38465TO 38861STEP22:POKEN,7:NEXT
69 IFC=1THENGOSUB80:C=0:GOTO10
70 GOTO100
80 PRINT"██DIN BLA BOLL I MITTEN JAGAS AV DATORNS RODA BOLL"
82 PRINT"█STYR UNDAN MED        JOYSTICKEN
83 PRINT"█NAR DU TRYCKER PA      SKJUTKNAPPEN SKICKAS  EN MALSOKANDE ROBOT    IVAG
"
85 PRINT"█START: TRYCK TANGENT█"
86 IFPEEK(197)=64THEN86
87 RETURN
100 GOSUB2000:Z=7680+X+22*Y:GOTO175
105 IFPEEK(197)=0 THENDY=-1:DX=0:GOTO175
```

```
110 IF PEEK(197)=56THENDX=+1:DY=-1:GOTO175
120 IF PEEK(197)=1 THENDX=+1:DY=0:GOTO175
130 IFPEEK(197)=57THENDX=+1:DY=+1:GOTO175
135 IFPEEK(197)=2THENDY=+1:DX=0:GOTO175
140 IFPEEK(197)=58THENDX=-1:DY=+1:GOTO175
145 IFPEEK(197)=3THENDX=-1:DY=0:GOTO175
150 IFPEEK(197)=59THENDX=-1:DY=-1:GOTO175
175 X=X+DX:Y=Y+DY
200 IFX=-1THENX=0:DX=-DX
205 IFY=-1THENY=0:DY=-DY
210 IFX=22THENX=21:DX=-DX
215 IFY=23THENY=22:DY=-DY
220 POKEZ,32
225 POKE7680+X+22*Y,81
230 POKEX+22*Y+38400,6
233 REM IF X>19ORX<2ORY>20ORY<2THENGOTO240
235 IF B=0THENB=1:GOTO100
240 B=0
300 IFQ8=0          THENS=1:Q=X:R=Y:Q8=1
310 IFS=0GOTO400
315 IFS>0THENPOKE7680+Q+22*R,32
325 IFQ<UTHENQ=Q+1
330 IFQ>UTHENQ=Q-1
335 IFR<VTHENR=R+1
340 IFR>VTHENR=R-1
345 IFS>1THENPOKE7680+Q+22*R,46
350 POKE38400+Q+22*R,0
352 S=S+1
353 IFS>8THENS=0
354 A=0
355 IFQ=UANDR=VTHENA=1:POKE7680+U+22*V,81:POKE38400+U+22*V,2
400 IFA=1THENA=0:GOTO100
403 POKE7680+U+22*V,32
404 IFU>XTHENU=U-1
405 IFU<XTHENU=U+1
410 IFV>YTHENV=V-1
415 IFV<YTHENV=V+1
420 IFU=-1THENU=0
425 IFU=22THENU=21
430 IFV=-1THENV=0
440 IFV=23THENV=22
445 POKE7680+U+22*V,81
450 POKE38400+U+22*V,2
451 REM FORT=1TO10:NEXT
452 IFU=XANDV=YTHEN500
455 GOTO100
500 T2=TI$/60
505 T2=T2-T1
510 PRINT "TID: " T2
515 IFPEEK(197)=32THEN010
520 GOTO515
2000 POKE37154,127:Q9=(PEEK(37137) AND28)OR(PEEK(37152)AND128):Q9=ABS((Q9-100)/4
)-7
2005 Q8=0:Q8=PEEK(37137)AND32:IFQ8THENQ8=1
2010 ONQ9GOTO 2100,2110,2120,,2130,2140,2150,,,,2160,2170,2180
2020 GOTO2100
2100 DX=-1:DY=+1:RETURN
2110 DX=-1:DY=-1:RETURN
2120 DX=-1:DY=+0:RETURN
2130 DX=-0:DY=+1:RETURN
2140 DX=-0:DY=-1:RETURN
2150 RETURN
2160 DX=+1:DY=+0:RETURN
2170 DX=+1:DY=-1:RETURN
2180 DX=+1:DY=+1:RETURN

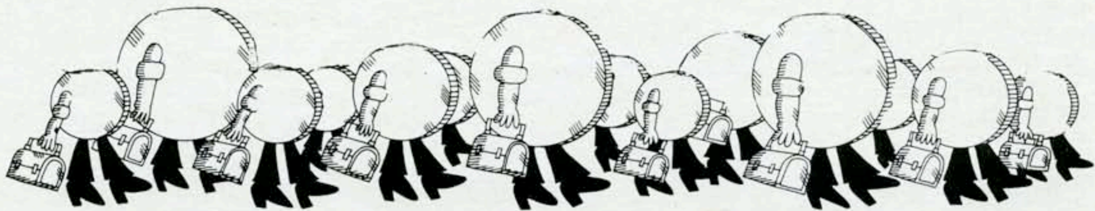
READY.
```

Planera din ekonomi

Flera läsare har hört av sig om ekonomiprogrammet i nr 5/6-83 och ville ha det på VIC-64. Vi bad Lars Moeschlin skriva om det för en 64:a och här det:

Programmet är uppbyggt med konton, där man lägger in sina fasta utgifter och inkomster. Vid avlöningstillfället ger programmet besked om hur mycket som skall sparas eller tas ut från bankkontot. Med de disponibla pengarna betalas räkningar och resten används för rörliga utgifter (mat, kläder etc).

Programmet nedan är avsett för VIC-64 och körs helt enligt inlagda instruktioner. För dig som har en VIC-20, finns programmet listat i VIC-rapport nr 5/6-83 och som vanligt kan den, som inte vill knappa in programmet själv, köpa en kassett eller disk med färdigt program direkt från författaren: Lars Moeschlin, S. Fiskebäcksvägen 84, 421 58 Västra Frölunda. Tyvärr kan han inte svara på telefonfrågor, så har du något du vill fråga om, var snäll och SKRIV till honom.



```
READY.

10 POKE53281,1
20 W=679:FORI=WTOW+35:READX:IFX<0THENPOKEI,PEEK(-X):GOTO30
25 POKEI,X
30 NEXT
40 DATA 169,-648, 162, 160, 32, 178, 2, 169
45 DATA 216, 162, 1, 134, 253, 133, 252, 169
50 DATA 0, 133, 251, 170, 168, 165, 253, 145
55 DATA 251, 200, 208, 251, 230, 252, 232, 224
60 DATA 4, 48, 244, 96
80 MK=50:NR=25:NC=40:BG=53281:NG=1:RG=6:GOTO100
90 GETX$:IFX$=""GOTO90
91 RETURN
100 ML=13:NP=12:MX=13:DIMT$(MK),B%(MK),S%(MK),H%(MK),V(ML),U(ML),P(ML),A(MX)
105 DIMR(NP),Q%(8):FORI=1TO8:READQ%(I):NEXT:G=214
106 DATA228,210,198,192,195,196,197,227
110 C$=CHR$(13):NK$=" NYTT KONTO ":GK$=" GAMMALT KONTO "
115 K$="KONTOHANTERING":R$="TIDIGARE SPARLöGE":L$="AKTUELLA KONTON"
120 S$="FRAMTIDA SPARLöGE":SK=55296:TK=1024
121 N$="QTRYCK NGN TANGENT"
130 PRINT"]QEEEKONOMISK PLANERING"
140 INPUT"QLAGRADE DATA IN";X$:IFLEFT$(X$,1)="N"GOTO220
150 INPUT"QFILNAMN";F$:OPEN1,1,0,F$:INPUT#1,NK,NL,NP,RM
160 FORK=1TONK:INPUT#1,T$(K),B%(K),S%(K),H%(K):NEXT
180 FORI=1TONL:INPUT#1,V(I),U(I),P(I):NEXT
185 FORI=1TONP:INPUT#1,R(I):NEXT:CLOSE1
190 PRINT"]Q  E*MENY*":PRINT"Q1 "K$:PRINT"Q2 KALKYLERING":PRINT"Q3 "R$
195 PRINT"Q4 "S$:PRINT"Q5 LAGRA"
200 INPUT"Q1,2,3,4 ELLER 5";X:IFX<1ORX>5GOTO190
210 ONXGOTO260,540,1000,952,1100
220 PRINT"]Q NY START":NK=0:NL=0:GOTO290
260 PRINT "]Q E";K$;"EEEEMENY■"
265 PRINT"Q1";NK$:PRINT"Q2";GK$:PRINT"Q3 LISTA KONTON":PRINT"Q4 "L$
270 PRINT"Q5 INGETDERA":INPUT"Q1,2,3,4EELLER5";X:IFX<1ORX>4GOTO190
280 ONXGOTO290,420,460,1200
290 NK=NK+1:K=NK:M=1
300 PRINT"]Q"NK$"NR="STR$(K):GOSUB310:GOSUB370:GOTO390
310 INPUT"QKONTONAMN";T$(K)
312 IFLEFT$(T$(K),1)<>"←"GOTO320
314 INPUT"QRADERA KONTOT";X$:IFLEFT$(X$,1)<>"J"GOTO320
316 FORI=K+1TONK:J=I-1:T$(J)=T$(I):B%(J)=B%(I):S%(J)=S%(I):H%(J)=H%(I)
```

```
318 NEXT:NK=NK-1:K=K-1:RETURN
320 INPUT"UTGIFT/PERIOD=";B%(K)
330 INPUT"STARTPERIOD=";S%(K):IFS%(K)>NPGOTO330
340 INPUT"PERIODINTERVALL=";H%(K):IFH%(K)>0THENRETURN
360 GOTO340
370 PRINT""RIGHT$(STR$(K),2)" "T$(K):PRINTSPC(6-LEN(STR$(B%(K))))B%(K);
380 PRINT"KR START"RIGHT$(STR$(S%(K)),2)" INT";
381 PRINTRIGHT$(STR$(H%(K)),2):RETURN
390 INPUT"KORREKT INSKRIVET";X$:IFLEFT$(X$,1)="N"THEN ONMGOTO300,440
400 PRINT"KONTOT LAGRAT":INPUT"FLER KONTON";X$
405 IFLEFT$(X$,1)="J"THENONMGOTO290,420
410 GOTO260
420 M=2:INPUT"KONTONR";K:IFK<1ORK>NKGOTO420
430 GOSUB370
440 PRINT"OM GAMLA DATA SKALL VARA KVAR TRYCK      <RETURN> EFTER ?"
445 PRINT"RADERA MED KONTONAMN ←"
450 GOSUB310:GOSUB370:GOTO390
460 K=1
470 PRINT"KONTOLISTA":J=0
480 GOSUB370:K=K+1:J=J+1:IFK>NKGOTO500
490 IFJ<6GOTO480
500 PRINTN$;
510 GOSUB90
520 IFK>NKGOTO260
530 GOTO470
540 PRINT"    KALKYLERING":M=3:IFNL=0THENINPUT"VILKET PERIODNR";PN:GOTO560
550 PN=P(NL)+1:IFRIGHT$(STR$(PN),2)>RIGHT$(STR$(NP),2)THENPN=PN+100-NP
560 PRINT"PERIODNR";PN:INPUT"KORREKT PERIODNR";Z$:IFLEFT$(Z$,1)="J"GOTO590
570 INPUT"VILKET PERIODNR";Z:IFNL=0THENPN=Z:GOTO560
575 IFZ=P(NL)THENM=4:PN=P(NL):PRINT"KALKYL FØR PERIOD";Z"GØRS OM":GOTO590
580 PRINT"KALKYLERING OMØJLIG":GOTO560
590 X=0:Y=0:FORI=0TONP:R(I)=0:NEXT:PRINT"":FORK=1TONK
595 PRINT"KONTO NR="K:I=S%(K)
600 Y=Y+B%(K):R(I)=R(I)+B%(K):IFI=VAL(RIGHT$(STR$(PN),2))THENX=X+B%(K)
610 I=I+H%(K):IFI<NP+1GOTO600
620 NEXT:RM=INT(Y/NP+.5):D=X-RM:PRINT"VERKLIG UTGIFT=";X
630 PRINT"MEDELUTGIFT=";RM:IFD<0THENPRINT"SPARA";-D;"KR
640 IFD>=0THENPRINT"TAG UT"D"KR
650 E=0:INPUT"EXTRA SPARANDE=";E:IFM=4GOTO680
660 IFNL<MLTHENNL=NL+1:GOTO680
670 FORI=2TONL:V(I-1)=V(I):U(I-1)=U(I):P(I-1)=P(I):NEXT
680 P(NL)=PN:U(NL)=U(NL-1)-D+E:PRINT"PLANERAT SPAR=";U(NL)
690 INPUT"VERKLIGT SPAR=";V(NL):PRINT"VERKL-PLAN="V(NL)-U(NL)"KR"
702 PRINT:PRINTN$
705 GOSUB90
710 GOTO900
720 X=10↑9:AA=-X:AB=X:IFAT=0THENAA=0:AB=0:M=2:E=NR-M-3
721 FORI=1TONA:IFAA<A(I)THENAA=A(I)
730 IFAB>A(I)THENAB=A(I)
740 NEXT:D=(AA-AB)/E:IFD<200THEND=200
750 Y=LEN(STR$(INT(D+1)))-2:Z=D/10↑Y:B=1:C=2:IFZ>1THENB=2:C=10
760 IFZ>2THENB=5
770 IFZ>5THENB=10:C=20
780 AD=B*10↑Y:CC=C*10↑(Y-3):L=-1:FORI=MTOE+M:LK=SK+NC*I:K=L:FORJ=5TO4+NA
785 K=NOTK:IFKTHENPOKELK+J,7
790 NEXTJ:L=NOTL:NEXTI:AI=AD*INT(AB/AD):AC=AI:FORI=E+M+1TOM+1STEP-1
800 X=INT(AC/200+.5)/5:IFX-CC*INT(X/CC)=0THENPOKEG,I:PRINT""XTAB(4)CHR$(239)
810 AC=AC+AD:NEXTI
820 FORI=1TONA:X=(A(I)-AI)/AD:K=INT(X):L=INT(8*(X-INT(X)))+1
830 POKETK+(E+M-K)*NC+I+4,Q%(L):NEXTI:RETURN
840 POKEG,NR-1:PRINT"ARTAL";:FORI=1TONL:X$=STR$(P(I)):X=VAL(RIGHT$(X$,2))
850 X$=MID$(X$,2,2):IFX<>12GOTO880
855 POKETK+(NR-1)*NC+4+I,231:IFI<3GOTO890
860 IFI<12ANDX>ITHENX=I
870 PRINTTAB(I-X/2+4)X$;:GOTO890
880 IFI=NLANDX>1ANDI>1GOTO860
890 NEXTI:RETURN
900 PRINT" PER  UTGIFT DIFFERENS":FORI=1TONP:X=R(I):Y=R(I)-RM
910 PRINTSPC(3-LEN(STR$(I)))ISPC(8-LEN(STR$(X)))XSPC(8-LEN(STR$(Y)))Y:NEXT
920 PRINTN$:GOSUB90
931 AT=0:NA=NP:FORI=1TONP:A(I)=R(I):NEXT:POKEBG,RG:PRINT"":SYSW
932 PRINT"TKR PERIODUTGIFT":GOSUB720
935 POKEG,NR:PRINTTAB(5)"123456789012 PER-IOD";
```

```
940 GOSUB90:POKEBG,NG:GOTO190
952 L=VAL(RIGHT$(STR$(P(NL)),2)):X=U(NL):A(1)=X:Y=0:FORI=2TONP:L=L+1
954 IFL>NPTHENL=1
960 A(I)=A(I-1)+RM-R(L):IFA(I)<XTHENX=A(I):Y=I-1
965 NEXT:PRINT"███"S$:PRINT"██BLIR MIN EFTER"Y"PER":PRINT"█MIN SPARAT ="X"KR"
967 PRINTN$:GOSUB90
970 POKEBG,RG:PRINT"█":SYSW:PRINT"██TKR "S$:AT=0:NA=NP:GOSUB720
975 POKEG,NR:PRINT"██EFTER01234567890█PERIO██████-DER█";
980 GOSUB90:POKEBG,NG:GOTO190
1000 PRINT"████"R$"██":PRINT" PER VERKL PLANE DIFF.█"
1010 FORI=1TONL:X$=RIGHT$(STR$(P(I)),4):Y$=RIGHT$(STR$(V(I)),6)
1020 Z$=RIGHT$(STR$(U(I)),6):A$=RIGHT$(STR$(V(I)-U(I)),6)
1030 PRINTSPC(4-LEN(X$))X$SPC(6-LEN(Y$))Y$SPC(6-LEN(Z$))Z$SPC(6-LEN(A$))A$:NEXT
1040 PRINTN$:GOSUB90
1050 POKEBG,RG:PRINT"█":SYSW:PRINT"██TKR VERKLIGT SPARLÖGE":AT=0:NA=NL
1051 FORI=1TONL:A(I)=V(I):NEXT:AT=0:GOSUB720:GOSUB840:POKEBG,RG
1060 GOSUB90
1070 POKEBG,RG:PRINT"█":SYSW:PRINT"█TKR VERKL-PLAN SPARL.":AT=1:FORI=1TONL
1080 A(I)=V(I)-U(I):NEXT:GOSUB720:GOSUB840:GOSUB90:POKEBG,NG:PRINT"█":GOTO190
1100 PRINT"██LAGRING AV DATA":X$=F$:INPUT"█FILNAMN";X$:IFX$<>""THENF$=X$
1105 INPUT"█BAND I KORREKT POS";X$:IFLEFT$(X$,1)<>"J"GOTO1100
1110 OPEN1,1,1,F$:PRINT#1,NK,C$;NL;C$;NP;C$;RM;C$;
1115 FORK=1TONK:PRINT#1,T$(K);C$;B%(K);C$;S%(K);C$;H%(K);C$;:NEXT
1120 FORI=1TONL:PRINT#1,V(I);C$;U(I);C$;P(I);C$;:NEXT
1130 FORI=1TONP:PRINT#1,R(I);C$;:NEXT:CLOSE1
1140 INPUT"█KORREKT LAGRAT";X$:IFLEFT$(X$,1)<>"J"GOTO1100
1150 END
1200 PRINT"███"L$:INPUT"VILKEN PERIOD";I:I=VAL(RIGHT$(STR$(I),2))
1210 PRINT"█ NR KONTO        BELOPP█":Y=0:FORK=1TONK:J=S%(K)
1215 IFI<>JGOTO1230
1220 Y=Y+B%(K):PRINTTAB(3-LEN(STR$(K)))KLEFT$(T$(K),11)TAB(21-LEN(STR$(B%(K))));
1225 PRINTB%(K)
1230 J=J+H%(K):IFJ<NP+1GOTO1215
1240 NEXT:PRINTTAB(8)"█SUMMA"TAB(21-LEN(STR$(Y)))Y:PRINT" "N$:GOSUB90:GOTO260

READY.
```





Rättelser

I VIC-rapport nr 2 har två fel smugit sig in på sidorna 4 och 18.
Vi publicerar därför dessa sidor igen med de rätta bilderna inlagda.

Markörförflyttningar

Genom att trycka på CRSR tangenterna på tangentbordet kan Du som Du sett, och säkert använt dig av åtskilliga ggr, flytta omkring markören på skärmen. Vad Du kanske inte vet är att du även kan använda dessa i dina PRINT kommandon. Detta Är mycket användbart i många fall och kan t ex vara nyttigt att kunna när man skall göra spel och liknande. Det är dock inte bara dessa tangenter Du kan använda utan även andra tangenter som påverkar skärmen på något vis går bra. Låt oss göra en lista över dem:

Shift + CLR/Home

Flyttar upp markören till översta vänstra hörnet efter att först ha raderat hela skärmen.

Inom situationstecken skrivs



CLR/home

Flyttar upp markören till översta vänstra hörnet av skärmen utan att radera den.

Inom situationstecken skrivs



INST/Del

Tar bort det tecken som står före markören.

Inom situationstecken skrivs



CRSR Upp

Flyttar markören uppåt en rad. Inom situationsteckens skrivs



CRS Ner

Flyttar ned markören en rad. Inom situationstecken skrivs



CRSR Vänster

Flyttar markören ett steg åt vänster. Inom situationstecken så skrivs



CRSR Höger

Flyttar markören ett steg åt höger. Inom situationstecken så skrivs





Med ÅÄÖ installerat

De tecken som skrivs ut när man trycker på en av dessa tangenter kallas KONTROLL TECKEN.

Användning av SHIFT + CLR/HOME har vi redan sett användas i de exempel vi tittat på i detta nr. Förutom att använda dessa tangenter så kan man också använda tangenterna för att byta färg och få reverserade tecken alltså dem man kan hitta under siffrorna. Testa dessa i olika printkommandon. Ha roligt! Du kan ju inte förstöra något, så vad gör det om något blir fel. Ha det så bra tills nästa månad då vi skall titta på nya SPÄNNANDE saker Du kan använda "VICKE" till.

```
5 PRINT"(SHIFT + CLR/HOME)(CRSR NER)(RVS ON)(22 blanktecken)(RVS OFF)"
10 PRINT"(SHIFT + CLR/HOME)"
20 FOR I = 1 TO 20
30 PRINT"(SHIFT + CLR/HOME)"
40 FOR J = 0 TO I
50 PRINT"(CRSR HÖGER)";
60 NEXT J
70 PRINT CHR (65 + I); :FOR K = 1 TO 90: NEXT:PRINT
80 NEXT I
90 PRINT"(SHIFT + INST/DEL)"
100 GOTO 5
```

Det som står inom parenteser skriver Du inte in i detta program utan *gör* det som står inom prentesen istället. T ex i fallet (SHIFT + CLR/HOME) så håller Du alltså SHIFT-tangenten nedtryckt medan Du trycker på CLR/-HOME-tangenten. Programmet presenterar inte särskilt mycket men kan, förhoppningsvis, introducera dig i markörförflyttningen värld.

Det enda du oftast får reda på när du börjar spela är att någon i spelet frågar dej . . .

— VAD SKA JAG GÖRA NU?

Sedan är det upp till dej att tala om vad som skall göras, och då har du som ända hjälp din egen fantasi samt de tillgångar spelet bjuder, i form av kommentarer ifrån ''Din utsände''. Vad han ser, hör och var han är.

Spelet bygger nästan bara på strängbehandling. Här hjälper ingen kunskap i supersnabba reaktioner med din väloljade joystick inte. Nej, bara din tankeverksamhet är väloljad så räcker det spelet igenom.

Hur som helst. Datorn förväntar sig alltså en mening ifrån dej, som t.ex. — TA LAMPAN, eller LÄS LAPPEN. När den fått en mening så behandlar datorn den på följande sätt. Först tar den reda på om det är ett eller två ord du skrivit, och det gör den genom att skilja orden åt vid mellanslaget. LÄS blir ORD 1 och LAPPEN blir ORD 2. Finns inget mellanslag blir allt du skrev ORD 1.

Det först ordet du skriver måste nästan alltid vara ett VERB, t.ex. TA, GÅ, ÖPPNA, SKJUT osv. Det andra ordet är antingen ett OBJEKT eller en riktning. Det vanligaste är att Adventures bara förstår två ord. Det går naturligtvis att få det att förstå hur många som helst, men det medför en desto krångligare programering, och längre. Jag kommer att till en början iallafall nöja mej med två.

Just den biten ser ut så här.

```
10 INPUT INS
20 FOR I=1 TO LEN(INS)
30 IF MIDS(INS,I,1)=" " THEN 50
40 NEXT: 01S=INS: GOTO 70
50 01S=LEFTS(INS,I-1)
60 02S=RIGHTS(INS,LEN(INS)-I)
70 PRINT 01S: PRINTO2S
80 GOTO 10
```

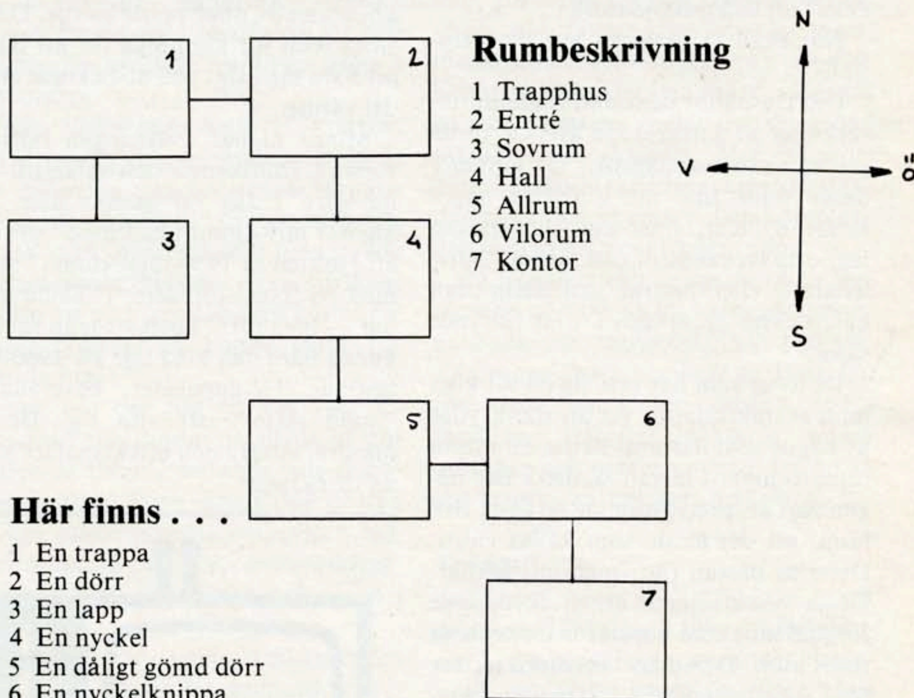
Prova så får du se. Men mera om programering och förklaringar till den kommer senare.

Men nu för att överhuvud taget kunna göra ett eget ADVENTURE eller ÄVENTYRSSPEL som det heter på Svenska, så krävs det en historia eller ett mysterium kring vilket spelet ska kretsa och gå ut på. Och till nästa gång tänkte jag inte sätta dej vid datorn och knappa, utan vid skrivbänken. Och sätta dom grå på prov, du ska nämligen skriva en sådan historia. Den kan handla om vad som helst, bara den har ett antal bestämda platser eller rum som du kan gå till, samt innehålla diverse prylar vilka ska bidra till mysteriets lösning.

Det behöver nu till en början inte betyda att du skenar iväg till något hundrafemtirumsslott med lika många gångar, vrår och prår, häxor, tomtar och elaka monster i drösar, utan jag tycker att du kan vara nöjd om du får ihop mellan 5 och 10 olika platser, och kanske lika många saker.

Det är ganska mycket att tänka på redan här, och jag tänkte visa dej hur jag brukar gå tillväga.

Rita en karta över platsen för skådespelet, gör en liten lista över de olika platserna samt sakerna. Den kan se ut t.ex. så här.



Allt det här kan självklart bytas ut mot sträckbänkar, piskor och dylika tingestar för dom som nu önskar det. Men vad det nu än vara må, så är principen alltid den samma.

Vidare så är det lämpligast att du går i väderstreckens riktningar, alltså nord, syd, väst och öst. Därför att GÅ BAKÅT eller GÅ HÖGER, blir bara bökigt. Det allra enklaste är också om man kortar ner alla riktningar man ska kunna gå åt, till bara en bokstav. Så ska du t.ex. här ovan från rum 5 till rum 6, så skriver du bara Ö. Sedan kan du även skriva ner alla kommentarer du vill ska förekomma i ditt spel, t.ex.

Datorn frågar.

— VAD SKA JAG GÖRA NU?

Du svarar.

— TA NYCKELN.

Datorn säger.

— DET FINNS INGEN HÄR!

Osv. Detta för att du ska få en så överskådlig bild av spelet som möjligt, innan du börjar göra program av det.

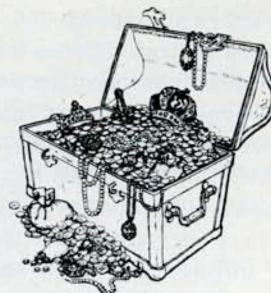
Det låter kanske både mycket och kanske krångligt, men när du kommit igång så flyter det både lätt och oftast alldeles för långt. Tänk bara som en dator när du skriver, INMATNING, BEARBETNING och UTSKRIFT. Men du måste ha med dessa kommentarer ifrån ''din man på plats'', därför att du dels måste vara i samma rum som de saker du vill ta, eller dörrar du vill öppna. Och du kan bara gå i de riktningar som rummen eller platserna tillåter.

Och grötar det ihop sig för dej så får du mera hjälp i kommande nummer.

Jag hoppas att jag förklarat mej på ett förståeligt sätt och att det låter intressant. Så vässa pennorna och spar oljan till cellerna, så kommer du nog alltid på något.

Lycka till, till nästa gång!

Börje Törnby



Hur gör man chips?

Commodore The microcomputer magazine, utgåva 27.

Commodores dotterbolag, MOS Technology Inc, producerar de chips som utgör hjärtat (och andra oumbärliga delar) till Din mikrodator.

Här skall vi beskriva hur de framställs.

Om Du tillhör dem som studerat tillverkning av integrerade kretsar (integrerad = sammanslagen), så kommer dessa sidor inte att innebära några större nyheter, men om Du, liksom jag, ofta tvekat inför ord såsom "halvledare", eller undrat vad detta som kallas chips egentligen är, läs följande sidor.

De av er som har gett sig på att kika inuti er mikrodator, vet att där kryllar av något som närmast liknar en massa tusenfotingar i metall. Kanske har någon sagt er, precis som jag en gång fick höra, att det är de som kallas chips. Detta är nästan rätt, men inte riktigt. Dessa tusenfotingar är en skyddande förpackning som kapslar in chipet som finns inuti. De kallas i tekniska termer för "DIL-kapslar" (Dual-In-Line-packages). På engelska säger man "DIPs".

När ett chip är monterat i en sådan kapsel med ben, kallas alltsammans för krets, IE-krets eller kapsel. Den del som Du inte ser, den ca 4x4 mm stora kiselbrickan inuti DIL-kapseln, innehåller den elektronik som Din dator är uppbyggd av.

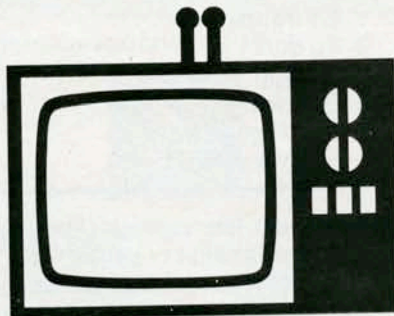
Det finns många olika typer av kretsar, var och en med olika funktion. Om en krets utför kommandon och utför beräkningar, kallar vi den för en mikroprocessor. Om den helt enkelt lagrar information, så är det en minneskrets. Om den kan få olika punkter på Din bildskärm att lysa, så är det en videokrets och om den styr högtalare, så är det en audiokrets.

Miraklet (som numera tas för givet) är att dessa komplexa kretsar, som för 35 år sedan tog hela rum i anspråk med en strömförbrukning som ett mindre samhälle, nu har reducerats till mikroskopiska streck på en liten bit kisel. Denna är dessutom inte större än en halv lillfingernagel och drar mindre ström än en liten ficklampa.

Kort historik

Många är väl ni (hoppas jag) som kommer ihåg när radio och TV-apparater av uppbyggda av rör. De var stora (och för klumpiga för att lätt gå att bära med sig) och utvecklade en hel del värme.

Minns ni när övergången började, först till transistorer och sedan till "solid state"? Jag var ganska ung, men jag har inte glömt hur "inne" det var, att i mitten av 1950-talet kunna "stila" med en transistorradio, ("Kolla grabbar — inga rör!") och att man faktiskt kunde *bära* den med sig! På 1960-talet började TV-apparater, baserade på "solid state", att visa sig. De var mindre, lättare och effektsnålare än sina föregångar.



Bakgrunden till den här utvecklingen hade lagts av två betydelsefulla uppfinningar: Transistorn 1947 och den integrerade kretsen (IC) 1960.

Båda utnyttjade halvledares egenskaper för att skapa elektriska komponenter för att först ersätta rören med transistorer och sedan åstadkomma kompletterа elektriska kretslösningar.

Låt oss för ett ögonblick stanna till här och säga några ord om halvledare och integrerade kretsar, eftersom de är så avgörande för hela den här utvecklingen mot datorer.

En halvledare är som namnet antyder ett material som normalt inte är en god elektrisk ledare såsom koppar, men inte heller en isolator, liksom glas eller gummi. Dock kan en halvledare på olika sätt försätter i det ena eller det andra tillståndet. Det är inte så konstigt som det låter.

Halvledare framställs genom att införa speciella störämnen (vanligtvis bor, arsenik eller fosfor) i extremt rent material, såsom kisel eller germanium. Tekniken att införa störämnen kallas "dopning". Störämnena, eller dopämnena, medför att det normalt oledande kislet, blir elektriskt ledande.

Om dopämnena kan införas på kiselbrickan med mycket hög lägesprecision, (jag skall förklara vad jag menar med precision längre fram) kan man hantera elektriciteten som man vill: förstärka den, bryta den och så vidare.

De mest strömsnåla kretsar som finns i dag är så kallade MOS-kretsar (MOS = Metal Oxide Semiconductor), och vad gäller transistorer talar man om fälteffekttransistorer av MOS-typ, så kallade MOSFETS. Dessa begrepp kan vara bra att känna till.

Integrerade kretsar, vilka är kompletta elektriska funktioner, gjorda av en bit halvledare, växte fram nästan samtidigt med halvledartekniken.

På en integrerade krets är alla ingående komponenter fabricerade på en enda bricka av helt rent kisel, (vanligast i datorsammanhang) ett så kallat substrat (underlag).

Den äldre motsvarigheten till IC-kretsarna utgjordes av kretskorten, som bestod av separata eller "diskreta" komponenter som motstånd, kondensatorer och transistorer (eller rör på gamla kretskort), ihoplödda på ett plastkort. I dag löder man ihop och förbinder IC-kretsarna med varandra på dessa plastkort.

Du kommer att märka en sak på en gång — integrerade kretsar kan vara mycket mindre. Med detta följer, för det första, att de är snabbare (kortare sträcka för strömmen att gå) och drar mycket mindre ström. De är också mycket billigare att producera. Och, allteftersom tillverkningsprocesserna fortsätter att förbättras blir de allt mindre — och billigare.

Jag ska inte gå in mer på detta, eftersom vi ska gå igenom MOS-teknologi lite senare. Men innan jag byter ämne, vill jag nämna att människor länge använde sig av halvledare

magnetband och skickas till det första steget i tillverkningsprocessen: fotomaskframställningen.

Innan jag går igenom fotomaskframställningen skall jag först säga några ord om wafers, eller skivor som är den svenska benämningen, efterson varje chip från början utgör en liten del av en stor kiselkiva.

Skivor är som namnet antyder, tunna, runda skivor av rent kisel. De framställs genom att smälta sand, att rena smältan från orenheter och att kristallisera det återstående kislet till en stav av enkristalligt kisel. Denna stav har för närvarande en diameter av fem till åtta tum (12,7—20,3 cm), men storleken ökar. Staven sågas upp i skivor om cirka en millimeters tjocklek och poleras spegelblanka innan de är mogna att få sitt kretsmönster på ovansidan.

På en skiva ryms cirka 400 chips.

I planarprocessen, vilken är den vanligaste framställningsmetoden för integrerade kretsar, beläggs skicorna med ett tunt skikt av metall eller kiseldioxid (som är en mycket god isolator) och mönstras med fotolitografiska metoder. Jag skall nu beskriva de olika stegen i planarprocessen.

Fotomaskframställning

När kretslösingen lämnar designavdelningen i form av ett magnetband, skickas den till maskframställningen. Här gör man ett fotonegativ av kretslösningen, som i storlek är cirka tio gånger så stor som den slutliga. Negativet görs på en fyra tums fyrkantig, ljuskänslig glasskiva, som kallas retikel. En krets byggs vanligen upp av flera lager, så man behöver en retikel för varje. I mera komplexa fall kan det röra sig om fyra till fem stycken.

Varje retikel används nu för att bygga den slutliga fotomaskinen för "sitt" lager i processen. Genom en annan fotografisk metod kan kretsmönstret på retikeln reduceras till sin slutliga storlek på cirka 4x4 mm. Detta lilla mönster reproduceras nu på en ny ljuskänslig glasskiva, och vi får en slutgiltig mask, som består av flera hundra likadana mönster för ett visst lager av kretsen.

Nu kommer vi till skivorna. Den slutgiltiga masken sätts in i en fotolitografisk maskin tillsammans med en skiva som belagts med ett ljuskänsligt skikt av så kallad fotoresist. När ljus passerar genom masken kommer exponerade delarna att härda. De delar som täcks av masken kommer att förbli oexponerade och härdar inte, varför de lätt kan avlägsnas med en kemisk lösning. Då fotoreisten avlägsnas, friläggs det underliggande skiktet som kan vara metall eller oxid, beroende på kretstyp. Tusentals wafers kan gå igenom denna behandling för varje lager i framställningsprocessen.

Som Du kanske redan förstått, kommer minsta störning i tillverkningen att få allvarliga konsekvenser och en dammpartikel är rena bombnedslaget. Därför sker större delen av processningen i så kallade rena rum, med noggrant kontrollerad temperatur (+ 1 C) och luftfuktighet. Dammpartiklar filtreras bort och personalen bär speciella skyddsoveraller, stövlar och huvudskydd. Till och med mustascher och skägg måste skylas — en perfekt arbetsmiljö för en allergiker.

Plasmaetsning, Diffusion och Jonimplantation

När fotoresisten på skivan har avlägsnats i den fotolitografiska processen kommer skivan i nästa steg att utsättas för ett plasma eller en exiterad kemisk ånga, för att uppnå en önskad effekt. Förr användes vätskor, men de områden som skulle etsas, blev till slut så små att vätskan inte längre kunde tränga in dit.

Nu är det dags att belägga skivan med de dopämnen som skall ge kiselytan sina elektriska egenskaper. De vanligaste dopämnena är som tidigare nämnts bor, fosfor och arsenik, och vilket som väljs och hur det appliceras på skivan beror på vilket lager man är i, samt vilka speciella kretsegenskaper man vill framställa.

Det finns två olika metoder för att applicera dopämnena: diffusion i värmeugn och jonimplantation.

Diffusion går till så att skivorna placeras i ett rack, så kallat skepp och förs med programmerad hastighet in i värme-ugnen, som håller cirka ett tusen grader C. De rätta kemiska gaserna blandas noggrant på ett bestämt sätt, varvid dopämnet tränger i, "impregneras" i kislet.

För nyare och mer komplexa kretsar är diffusion inte alltid den bästa metoden och i dessa fall används jonimplantation.

Metoden innebär som framgår av namnet, att dopämnena joniseras vartefter de accelereras mot ett elektriskt fält. Strålen får träffa kiselytan, varvid dopämnena tränger in till önskad mängd och önskat djup. Exakt rätt mängd och rätt djup är av oerhört stor betydelse för att kretsen skall fungera.

Slutfas och kontrollmätning

Efter cirka tre veckor är skivan klar efter att ha slussats ut och in i olika processteg. Skivan är nu täckt med ett skyddande lager av kiselnitrid och genom ytterligare en fotolitografisk process så friläggs de något större fyrkantiga metallytorna, de så kallade bondytorna, dit anslutningstrådarna fästs eller bondas.

Innan skivorna sågas itu till enskilda chips så skall de kontrollmätas. Det sker med hjälp av ett antal nålfina mätprobar som sänks ner mot skivan och mäter med hjälp av en dator på varje chip för sig. Om ett chip är felaktigt så markeras det med bläck. I de flesta industrier i dag är cirka 50 procent av chipen felaktiga, så det gäller att sålla bort de dåliga i god tid.

Kapsling

Slutligen sågas skivan till enstaka chips med hjälp av en mycket fin diamantsåg innan chipet monteras i en DIL-kapsel. Kretsen ansluts till benen på kapseln via mycket tunna guldtrådar. Detta kallas bondning och görs under mikroskop. (Om Du bryter upp en gammal kapsel kan Du se trådarna med förstoringsglas.)

Till sist sätts locken, av plast eller keramik, på och vi har fått en färdig krets, klar att sättas in i Din dator, eller i en diskmaskin.

Jag hoppas de inte blandar ihop chipen, för i så fall kanske vi får upproriska diskmaskiner, som spelar oss spratt och räknar ut vattenförbrukning i stället för att diska.

Det är en liten klyscha, men jag tänker säga den ändå; I framtiden kan ni nog räkna med att IC-kretsar kommer att bli ännu mindre och ännu mer komplexa. Vi kommer att kunna uträtta mer med lägre strömförbrukning till ett lägre pris.

Nya upptäckter som ökar fotografisk upplösning fortsätter att påverka framställningsprocesserna. Storleken

bestäms i alla fall till viss del av upplösningen vid fotomaskframställningen.

När datorer gör datorer kan effekterna bli slående. Den sortens gissningar kan ni läsa överallt. Jag har i första hand velat ge er en del fakta.

Förhoppningsvis behöver ni i fortsättningen inte känna er helt borta när era tekniskt intresserade kompisar börjar prata om halvledare, integrerade kretsar och chips.

REFERENSER

National Geographic, oktober 1982, innehåller en mycket detaljerad, men för den skull lättläst förklaring av mikrochipstillverkning. Artikeln vänder sig till lekmän.

Scientific American, september 1977, ger en mera tekniskt ingående, och inte fullt lika lättläst, men mycket nyttig genomgång av mikorelektronik. Artikeln vänder sig i första hand till lite mer insatta personer.

Technology Illustrated, februari/mars 1982, presenterar något ytliga, men trots detta ganska bra upplysningar om chips.

Workman Publishing's: 1984 Computer Diary, är ett ställe där Du kan hitta en hel del smått och gott om datorer. T ex ursprunget till ett ord som "robot", och fotografier av ENTAC.

För er som vill ha ännu mer, leta efter Reader's Guide to Periodical Litterature, från 1972 och framåt. Speciellt intressanta är de artiklar som stod att läsa i "Popular Electronics" och "Popular Science". Vissa av dem ger såväl ett gott bakgrundsmaterial som ett gott skratt.

Tekniska begrepp och fakta

Bi-polar transistor
En transistor som framställts att sammanfoga tre skikt, varav det mittersta är positivt dopat, medan de två yttre är negativt dopade, eller vice versa.

CAD (Computer Aided Design)
Används som hjälpmdel för att designa och testa integrerade kretsar. Detta möjliggörs bl a genom att jämföra data angående ett nytt chip's funktion med information från gamla kretsar, som finns lagrade i en databas.

Chip
Smeknamn för en integrerad krets, som mönstrats på en liten bit kisel. Ett svenskt, men sällan använt ord är bricka.

Diffusionshammare
En värmeugn med hög temperatur används vid chipsframställning. Där diffunderar dopämnen in vissa utvalda delar av kretsmönstret för att få kislet halvledande.

DIL (Dula-In-Line)
Den kapsel som chipet sitter monterat i liknar närmast en tusenfoting i metall. Benen står i två rader, varav namnet DIL. Denna går under DIL-kapsel och tillverkas i plast eller keramik.

Dopning
Process, där utvalda störämnen (kallas dopämnen) såsom fosfor, bor och arsenik, tillförs ett rent material som kisel eller germanium, för att skapa en halvledare.

IC (Integrerad circuit)
På svenska: integrerad (= sammanslagen) krets. Här avses en kiselbricka med ett pålagt, diffunderat kretsmönster. I Sverige används oftast ordet "IC-krets" för DIL-kapsel plus chip, ihopmonterade. Ibland används även "DIL-kapsel", "krets" eller "kapsel" för samma sak. Även monolitkrets (monolit = en sten, d v s kiselbit i detta fall) förekommer.

Jonimplantation
En mycket exakt metod för att införa dopämnen i kislet. Enstaka joner "skjuts" in i kislet under noggrant kontrollerade former.

LSI (Large Scale Integration)
Betecknar IC-kretsar med 40 000 till 100 000 transistorer.

Mikroprocessor
En IC-krets, avsedd för att utföra kommandon och göra beräkningar.

MOSFET (Metal Oxide Semiconductor Field Effect Transistor)
En transistor som formats genom att skapa "öar" av negativt och positivt dopade områden. Dessa förenas genom en kanal av kiseldioxid med ett metallager ovanpå. Genom att lägga en spänning på metallen kommer ström att flyta i kanalen nästan proportionellt mot den pålagda spänningen.

Fotomask
Används i den fotolitografiska processen vid chipframställning. Den skyddar (maskar av) de områden som inte bör exponeras, så att de senare kan etsas.

Planarprocessen
En vanlig framställningsprocess för integrerade kretsar. Kiselskivan beläggs med ett tunt lager av kiseldioxid, som sedan mönstras med fotolitografisk teknik, för att frilägga vissa ytor, där dopämnen förs in.

Plasma-etsning
En del i framställningsprocessen. Efter det att en skiva har genomgått den fotolitografiska delen så etsas vissa ytor med hjälp av en exiterad kemisk ånga, som kallas plasma.

Retikel
Ett fotonegativ av ett lager av kretsen. Retiklar vid MOS Technology Inc, görs på glas som beläggs med en ljuskänslig emulsion. Storleken är cirka tio gånger större än den slutgiltiga kretsen.

Halvledare
Ett material som är en sämre elektrisk ledare än koppar, men bättre än glas. En halvledares elektriska egenskaper kan ändras för att tillfredsställa olika behov. På engelska: semi-conductor.

Kisel
Ett ämne som används i ren kristallisk form som grundmaterial för många typer av integrerade kretsar.

På engelska: silicon. Därav det kända namnet Silicon Valley.

Sincap
Den sista skyddande hinnan över en färdig kiselskiva. Hinnan består av kiselnitrid, som skyddar mot mekanisk och kemisk påverkan.

Solid state
Elektriska signaler och funktioner (t ex switching) sker i ett helt fast medium, det vill säga, ett chip av kisel.

Transistor
En halvledarkomponent som fungerar antingen som förstärkare eller som strömswitch.

VLSI (Very Large Scale Integration)
Integrerade kretsar som rymmer mer än 100 000 transistorer.

Skiva
En tunn skiva av kisel, på vilken man mönstrar några hundra likadana chips. En skiva har en typisk storlek på fem tum. På engelska: wafer.

Yield (utbyte)
Andel fungerande kretsar av totalt antal framställda. Typiskt värde kan vara 50 procent. Alla fabrikanter är mycket förtegna om hur bra yield de har.

Viktigt årtal i mikrodatorns historia

1642
En utväxlad additionsmaskin uppfanns av den franske matematikern Blaise Pascal (endast 19 år gammal). Gottfrid Leibniz förbättrade konstruktionen senare.

1820
''Analysmaskinen'' konstruerades av Charles Babbage. Den hade ett minne som kunde hålla 1.50 nummer och en ''kvarn'' för att utföra aritmetiska operationer. Han föreslog att data skulle matas in via hålkort, liknande de som användes i de för den tiden avancerade vävmaskinerna. Trots att analysmaskinen var för mekaniskt avancerad för att kunna konstrueras, så räknas Babbage fortfarande som datorålderns fader.

1880
Ett snabbt hålkortsbaserat datasystem uppfanns av Herman Hollerith för att hjälpa de amerikanska myndigheterna vid folkräkningen 1880. Han banade vägen för elektronisk databehandling.

1884
''Comptometer Business Machine'' byggdes av Dorr E Felt med hjälp av ???skruvar, gummiband och en makaronilåda i tr. Den klarade multiplikation och division för allmänt bruk, och blev till sist en av de mest betydande kontorsmaskinerna vid 1900-talets början.

Tidigt 1900-tal
Första bruket av kristallradiomottagare. (Tidiga spetskontakt dioder av halvledarmodell.)

1902
Elektronröret konstrueras av Ambrose Flemming.

1944
Den första riktiga datamaskinen: Mark I från Harward. Mark I var en sifferkalkylator för allmänt bruk som utvecklats i samarbete med IBM. En av de konstruktörerna Howare Aiken, sade: ''Babbages dröm har förverkligats''.

1945
Den första elektroniska digitala datorn ENIAC (Electronic Numerial Integrator and Calculator). Den vägde 30 ton, upptog ett rum på 10x16 meter och innehöll 18 000 rör. Det enda sättet att ändra programmet var att koppla om ledningarna. Konstruktörerna trodde att fyra sådana maskiner skulle räcka för hela USA's samlade behov av datorkraft för beräkningar.

EDVAC (Electronic Discrete Variable Automatic Calculator) introducerades av John von Neuman vid Princeton-universitetet. EDVAC var den första datorn med switchar av on-off-typ för att representera instruktioner och data, samt att låta både data och program att lagras i minnet.

1947
Den första kommersiella halvledarkomponenten — bipolartransistorn — utvecklas av William Shockley, Walter Brattain och John Bardeen vid Bell-laboratorierna.

1951
Tryckta kretsar utvecklas av Danho och Abrahamsson vid USA's signaltrupper.

Det första kärnminnet patenteras av Jay Forrester. Detta innebär snabb tillgång (access) på data och möjligheter att lagra stora informationsmängder.

1952
Genombrott för framställning av integrerade kretsar nåddes av G N A Dummer vid Royal Rader Establishment i England.

1955
De första framställda och testade fälteffekttransistorerna. Osäker framställningsteknik förhindrade massproduktion.

1956
Första datorn för självbygge släpptes ut av Heath Company. Priset var USD 700.

1957
Planarprocessen utvecklas av Robert Noyce och Gordon Moore, vid Fairchild Semiconductor. Grunden hade lagts av Frosch och Derrick m fl genom arbeten vid Bell-laboratorierna.

1959
Utveckling av integrerade kretsar i form av chip vid Fairchild Semiconductor och Texas Instruments.

1961
Robert Noyce belönades med patent för den integrerade kretsen. Händelsen undgick världens ögon.

1962
Första fungerande fabrikstillverkade MOSFET-transistorn (Metal

Oxide Field Effect Transistor) och utveckling av integrerade kretsar av MOS-typ).

1967
De största problemen med att framställa MOSFET-transistorer löses, vilket resulterar i ökat processutbyte och sänkta kostnader.

1968
Först RAM-minnet av MOS-typ med plats för 64 till 256 bitar.

1970
Första kalkylatorchipen av MOS-typ. Första 1-kbit (ungefär 128 bytes) fullt avkodade RAM-minnet presenterades.

1971
Första CPU på ett chip utvecklas av Ted Hoff vid Intel. Detta jämnade vägen för de första mikrodatorerna som kom kort därefter.

1972
Första populära artikeln om mikrodatorer kom i Business Week den 12 maj; ''Microcomputers aim at a huge new market''.

1970
Mikroprocessorn 6502 utvecklas av MOS Technology Inc.

1975
Commodore köper MOS Technology Inc.

1977
De första PET-datorerna släpps ut på marknaden med 8k RAM-minne.

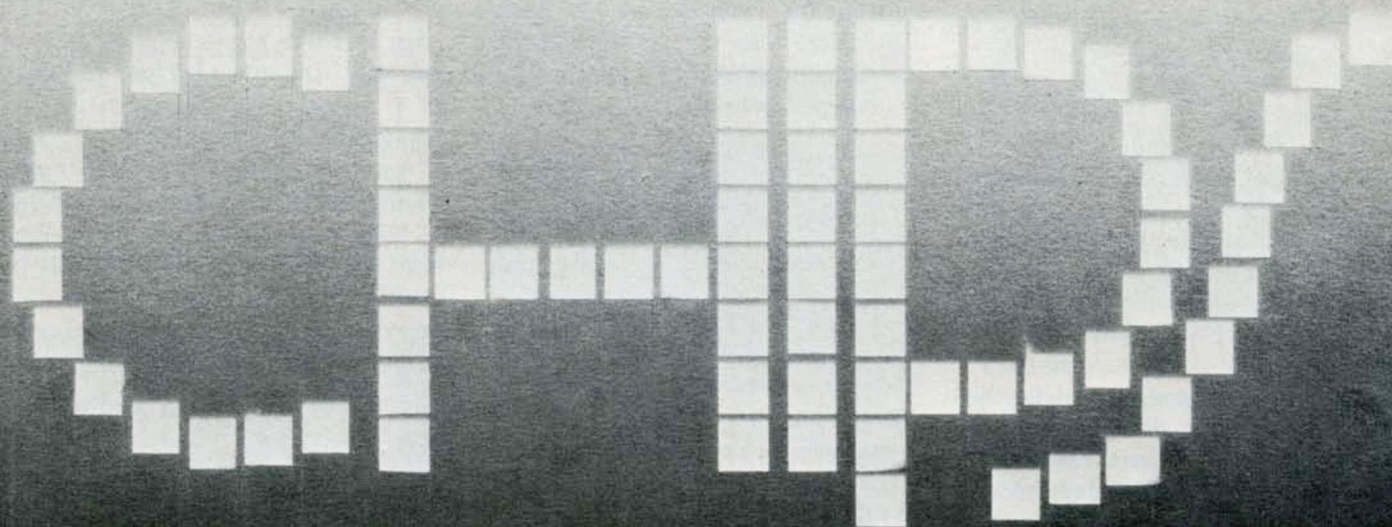
1980
Förbättrade framställningsprocesser möjliggör tillverkning av chips med mer än 60 000 transistorer.

1981
Den första billiga och helt kompletta hemdatorn, VIC 20, introduceras av Commodore.

1982
Commodore 64, med fler funktioner än många dyra kontorssystem, släpps till ett lågt pris.

1983
Mikrodatorer blir billiga nog att vara ''var mans ägo'', och lika vanliga som många andra hushållsprylar.

1984
Det bästa återstår ännu att se.



Assemblerskolan

Då var det dags igen för lite assemblerprogrammering. Från och med detta nummer så kommer jag, Åke, att ensam svara för denna serie. Jag hoppas att detta inte skall innebära några negativa konsekvenser för serien, men måste direkt konstatera att eftersom jag inte har tillgång till en 64'a så måste alla programexempel fortsättningsvis vara inriktade på ''lillebror'' vic-20. Programexemplen kommer dock att ligga med början på S 1200 och detta betyder att även Du som har en 64'a kan köra dessa i de flesta fall. När det är uppenbart att ett exempel inte går, eller snarare inte fungerar på 64'an, så kommer jag att påpeka detta. För måga kan kanske detta att samma maskinkodsprogram kan köras på vic-20 och vic-64 verka en aning konstigt då det sitter två olika processorer i de två. Den ena heter 30 6502 (i vic-20's fall) och den andra 6510. Sanningen är att det inte är några skillnader när det gäller maskinkodsprogrammering mellan dem.

6510 har dock de tre lägsta adresserna på sida noll ockuperad av register för 1/0 port som ingår i chipet. D v s det man bör titta närmare på när man vill konvertera ett program från 6502'an till 6510'an är de adresseringar som görs på ''zero-page'' adresserna S00 — S02 då dessa i ett normalt 6502 system är RAM. I denna serie så kommer vi att i möjligaste mån att försöka undvika dessa minnespositioner.

Då kastar vi oss på själva programmerandet.

I den senaste artikeln så tittade vi på hur man laddade ackumulatorn på några olika sätt. I detta nummer så skall vi kolla hur vi får in data i de s k indexregistren samt hur vi ''kastar'' ut denna information till minnet igen. Vi lärde oss tre adresseringsmeoder i förra avsnittet nämligen Immediate, zero-page och absolut. För informationens skull kan jag nämna att det existerar 13 st totalt, men att vi så här till en början begränsar oss till dessa tre och berör de återstående när vi får användning för dem.

Att ladda indexregistren med data är inte svårare än att ladda ackumulatorn. De mnenomics vi använder oss av är
LDX
och
LDY
där LDX säger att vi skall ladda X-registret med data och LDY är motsvarande instruktion för att ta in till Y-registret. Låt oss nu plocka fram maskinspråksmonitorn och slå in följande rad

```
. : A2 10 A6 C0 AE
. : 0005 00 20 A0 10 A4
. : 120A C0 AC 00 20 00
```

För att slå in raderna så skriv först
M 1200 <RETURN>
så sköter monitorn om att lägga in ''nuffrorna'' i minnet, och ger dig automatiskt en indikering på i vilken adress Du håller på att ''greja'' med. När alla tre raderna är inskrivna så slår Du return två ggr och vipps så är Du ute i monitorns kontrollslinga.

Skriv nu
D 1200 120E (glöm ej mellanslagen)
så får Du se hur den MASKINKOD Du nyss slog in ser ut i sin ASSEMBLERFORM. Om du har slagit in det hela riktigt (och tryckfelsnisse har hållt sig undan) så bör det se ut så här:

```
. , 1200 LDX # $10
. , 1202 LDX $C0
. , 1204 LDX $2000
. , 1207 LDY  #  10
. , 1209 LDY $C0
. , 120B LDY   $2000
. , 120E BRK
```

Detta program !!! uträttar inte så mycket men du kan ju ändå skriva
G 1200
för att se vad som händer. När Du alltså programmet och när detta når BRK instruktionen så hamnar vi i Monitorn och registerinnehållet i processorn skrivs ut. I fallet VICMON så bör det se ut ungefär så här:

```
B* PC   SR  AC  XR  YR  SP
.;120E  B1  00  FF  FF  F6
```

Det är bara kolumnen XR och YR som vi är intresserade av och har Du fått något annat än $FF i dessa kolumner så beror det helt enkelt på att minnespositionen $2000 innehåller något annat än $FF (den nyfikne rackaren kan ju skriva M 2000 <Return> och se om det stämmer med det som ligger i X och Y registren).

Från det Du slog in ovan kan Du lätt kontrollera vilken OPCODE som LDX och LDY har i de olika adresseringsmoderna. För den late så listar vi dessa nedan

LDX Immediate	$A2
LDX Zero-page	$A6
LDX Absolut	$AE
LDY Immediate	$A0
LDY Zero-page	$A4
LDY Absolut	$AC

I slutet av denna artikel så hittar Du tabell 1 som förutom en beskrivning av varje MNEMONICS ger information om den tid det tar att utföra varje instruktion, antal byten den kräver och vilka adressmoder som för den aktuella instruktionen. Bekymra dig inte ännu över de adressmoder som vi inte har berört, dessa skall vi titta på i sinom tid.

Jag tror inte att vi behöver gå igenom vad LDX och LDY gör förutom det vi redan diskuterat, eftersom de har fullständigt analog funktion med LDA som vi beskrev i förra artikeln. Låt oss istället gå vidare och titta på hur man lägger ut data från ett register till en minnesposition. För detta har vi tre instruktioner:

STA ''STore Ackumulator in memory''
= Lagra ackumulator i minne.
STX ''STore X-register in memory''
= Lagra X-register i minne.
STY ''STore Y-register in memory''
= Lagra Y-register i minne.

Vad var och en av dessa åstadkommer är väl kanske klart men låt oss ändå titta lite närmare på STA (STX/STY är helt analoga med STA).

När vi skriver t ex
STA $2000
så tar processorn innehållet i ackumulatorn och flyttar det till minnescell $2000. Man måste nu fråga sig VAD LIGGER I ACKUMULATORN EFTER DET ATT INSTRUKTIONEN HAR EXIKVERATS (körts)?

Låt oss studera detta genom att slå in följande lilla programsnutt.

1200 LDA # $ØF Ladda ackumulatorn med $ØF.
1202 STA $2000 Lagra ackumulatorns innehåll i $2000, d v s ($2ØØØ).
1205 BRK ;, Klart! Gå tillbaks till monitor.

(Om du inte vet hur Du skall slå in detta så se föregående artikel, i fortsättningen förutsätter jag att du kan A-kommandot på din monitor).

Ta nu och kör detta program. (G 1200 <RETURN>)

När programmet är kört så bör Du få något i den här stilen på skärmen:

B * PC SR AC XR YR SP
.; 1205 31 OF 63 00 F6

Under kolumn AC så ser Du att vårt $OF ligger. (Bekymra dig inte om dom andra kolumnerna dom är med största säkerhet inte desamma på din dator).

Detta betyder att $OF fortfarande ligger kvar i ackumulatorn efter det att vi har kört
STA $2000

Vi kan alltså dra slutsatsen att STA KOPIERAR ackumulatorns innehåll ned till den minnesposition som vi vill lagra dess innehåll i. D v s vi förstör ingen data i våra register genom en dylik instruktion. (Den nyfikne kollar lämpligen med hjälp av M kommandot att SOF verkligen ligger i $2000).

Denna egenskap är något som gäller generellt för alla instruktioner och är mycket viktig att notera. Instruktionerna verkar på ett register eller en minnesposition men de verkar på ett sådant sätt att de ej förstör det tidigare datainnehålet. Naturligtvis är detta sant men man måste också komma ihåg att man t ex med
LDA $20
ersätter det som tidigare låg i ackumulatorn med *innehållet* i minnesposition $20 men att innehållet i $20, vad det nu än var, blir oförändrat av operationen.

STX och STY har exakt samma funktion som STA (de lagrar naturligtvis(x) respektive(y) och de adressmoder som existerar för de tre hittar Du som vanligt i tabell 1. Obsevera dock att den adressmod som vi kallar IMMEDIATE in-te finns här. Den skulle ju vara ganska meningslös också i detta fall eftersom den kräver att den data den skall verka på ligger efter opcoden. Låt oss nu återigen titta på ett litet och ganska ''löjligt'' program exempel där vi använder oss av det vi har lärt oss.

Uppgiften är så enkel som att skriva VIC på skärmen. Detta skulle vi kunna göra genom att lägga några byte i minnet på ''Vicke'' och sedan fylla på med färg. I själva operativsystemet finns dock en SUBRUTIN som sköter detta åt oss och som har startadress $FFD2 (gäller även 64'an).

Genom att anropa denna subrutin så kan vi få det tecken som ligger i ackumulatorn (vi-d anropet) att printas på skärmen. För att anropa en subrutin så använder vi en mnemonic på formen JSR. Vi bekymrar oss inte nu för hur denna fungerar utan spar detta till ett senare tillfälle. Vi måste dock vet att efter det att subrutinen har körts så fortsätter programmet med den instruktion som ligger efter JSR. Den rutin som ligger på SFFD2 kräver att det tecken som ligger i ackumulatorn är kodat i den s k ASCII-koden. Hur denna kod ser ut kan Du se i tabell 2.

Från denna tabell så kontrollerar vi vilka koder som behövs för att skriva

''VIC''
''V'' = $56
''I'' = $49
''C'' = $43
1200 LDA # $56
; Ta in koden för ''V''.
1202 JSR $FFD2
; Printa ''V'' på skärmen.
1205 LDA # $49
; Ta in koden för ''I''.
1207 JSR $FFD2
; Printa ''I'' på skärmen.
120A LDA # $43
; Ta in koden för ''C''.
120C JSR $FFD2
; Printa ''C'' på skärmen.
120F BRK ; Klart

Kör programmet och ''abracadabra'' så skrivs ''VIC'' ut på skärmen. Jag rekommenderar att Du stannar till här och skriver ut egna saker på skärmen med hjälp av den teknik som vi använt i programmet ovan. Man måste direkt konstatera att det finns betydligt bättre sätt att skriva ut en sträng på skärmen, (en sträng är en följd av bokstäver,) och inom sinom tid så skall vi gå igenom även sådana metoder. Något annat Du absolut inte skall ta efter är mitt sätt att kommentera programraderna. Kommentarerna skall beskriva *hur PROGRAMMET fungerar* inte hur ENSKILDA INSTRUKTIONER fungerar för man får förutsätta att om någon annan skall läsa en assemblerlistning så är denne någon insatt i assemblerprogrammering. Som du förstår så kan man lätt bli irriterad på att i en och samma programlistning få lära sig hur LDA fungerar på varannnan rad. Jag däremot har fått dispans av mig själv för att riktigt tjata in instruktionerna i huvudet på er som följder denna serie.

Något som också varmt rekommenderas för alla programexempel är att använda W-kommandot på din monitor och stega dig igenom programmet för att följa exikveringens gång. HESMON har en unik egenskap i denna mod som möjliggör att man hela tiden följer registerinnehållet. Tyvärr saknas denna egenskap på VICMON men man kan ju istället närsomhelst avbryta stegningen och kontrollera registerinnehållet med R-kommandot.

Låt oss nu titta på ett par instruktioner till.

Något som skulle vara mycket användbart vore att kunna överföra data från ett register till ett annat. På 6502'an (6510) så finns fyra instruktioner för att utföra detta.

TAX ''Transfer akumulator to X-register'' = kopiera (A) till X-register.
TAY ''Transfer akumulator to Y-register'' = kopiera (A) till Y-register.
TAY ''Transfer X-register to Akumulator'' = kopiera (X) till akumulatorn.
TYA ''Transfer Y-register to Akamulator'' = kopiera (Y) till akumulatorn.

Kom ihåg att (X) betyder att det är innehållet som avses, i detta fall är det alltså innehållet i X-registret.

Vad dessa instruktioner gör är alltså bara att *kopiera* över innehållet från ett register till ett annat. Detta kan vara speciellt lämpligt när man t ex vill ladda in data till ett av indexregistren med en adressmod som bara existerar för LDA (se tabell 1). Det händer också relativt ofta att man behöver lagra data temporärt i ett av indexregistren medan man jobbar med annan data i akumulatorn. Vi kommer att se många fler exempel på detta längre fram. För att se hur instruktionerna fungerar så slår vi in nedanstående program.

```
1200    LDA # $41
 ;  Ta in koden för "A'.
1202    LDX # $42
 ;  Ta in koden för "B".
1204    LDY # $43
 ;  Ta in koden för "C".
1206    JSR $FFD2
 ;  Skriv ut "A" på skärmen.
1209    TXA
 ;  (X)→ A
120A    JSR
 ;  Skriv ut "B" på skärmen.
120D    TYA
 ;  (Y)→ A
120E    JSR $FFD2
 ;  Skriv ut "C" på skärmen.
1211    BRK
 ;  Klart!
```

Vad programmet gör och hur det fungerar behöver väl knappast förtydligas men det finns en viktig egenskap hos denna subrutin (FFD2) som måste noteras. Tänk dig att det program som vi skrev ovan skulle användas som en subrutin. Varje gång vi anropade denna rutin så skulle vi då få "ABC" utskrivet på skärmen. Eftersom vår rutin använder både ackumulatorn och de två indexregistren så sklle den data som vi hade i dessa innan vi anropade subrutinen försvinna när vi återvänder. Som Du ser ovan så förstörs inte denna data av "print subrutinen". Detta betyder bara att denna rutin spar undan indexregistrens innehåll när vi kallar på den och sedan åter lägger tillbaks dessa innan den återvänder till vårt program. I de flesta subrutiner så sker ej detta och man måste därför tänka på vad man har i sina register innan man anropar en viss rutin. I den programmerings handbok som finns att tillgå till VIC-20 kan Du hitta fler rutiner som Du kan anropa från dina egna program och som i mångt och mycket underlättar din programmering. Du kommer även att på annat ställe i denna tidning hitta sådana beskrivningar under våren. Så mycket mer hinner vi nu inte med denna gång utan får träna och stå i till nästa månad då det kommer nya frächa instruktioner att svettas eller glädjas över.

Mnemonic: LDX

Beskrivning: X-registret laddas med data i en minnesposition

Flaggor:

N	V		B	D	I	Z	C
●	–					●	

Adressmod	kod	antal bytes	cykler	"X"	exempel
Absolut	$ AE	3	4	Ø11	LDX $ 2ØØØ
Zero-page	$ A6	2	3	ØØ1	LDX $13
Immediate	$ A2	2	2	ØØØ	LDX # $FF
Absolut, Y	$ BE	3	4*	111	LDX$12ØØ, Y
Zero-page, Y	$ B6	2	4	11Ø	LDX $ Ø3, Y

* Dessa tar en cykel extra att utföra om page passeras.

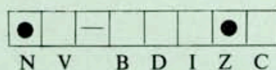
Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte

7	6	5	4	3	2	1	Ø
1	Ø	1	×	×	×	1	Ø

Mnemonic: LDA
Beskrivning: Akumulatorn laddas med innehållet i en minnesposition.

Flaggor:

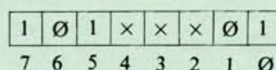


Adressmod	kod	antal bytes	cykler	"X"	exempel
Absolut	$ AD	3	4	Ø11	LDA $ 2ØØØ
Zero-page	$ A5	2	3	ØØ1	LDA $ 13
Immediate	$ A9	2	2	Ø1Ø	LDA # SFF
Absolut, X	$ BD	3	4*	111	LDA $ 12ØØ,X
Zero-page, X	$ B5	2	4	1Ø1	LDA $ CC,X
Absolut, Y	$ B9	3	4*	11Ø	LDA $ CØØØ, Y
(Indirekt, X)	$ A1	2	6	ØØØ	LDA ($ Ø1,X)
(Indirekt), Y	$ B1	2	5*	1ØØ	LDA ($ 2Ø), Y

* Dessa tar en cykel extra att utföra om page passeras.

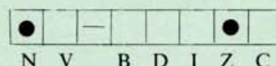
Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte



Mnemonic: LDY
Beskrivning: Y-registret laddas med data i en minnesposition.

Flaggor:

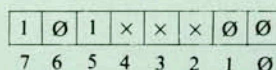


Adressmod	kod	antal bytes	cykler	"X"	exempel
Absolut	$ AC	3	4	Ø11	LDY $ 2ØØØ
Zero-page	$ A4	2	3	ØØ1	LDY $ CØ
Immediate	$ AØ	2	2	ØØØ	LDY # SFF
Absolut, X	$ BC	3	4*	111	LDY $ 12ØØ, X
Zero-page, X	$ B4	4	4	101	LDY $ Ø3, X

* Dessa tar en cykel extra att utföra om page passeras.

Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte



← bit

Mnemonic: STA
Beskrivning: Akumulatorns innehåll kopieras till en minnesposition.

Flaggor:

N	V		B	D	I	Z	C
		–					

Adressmod	kod	antal bytes	cykler	"X"	exempel
Absolut	$ 8D	3	4	Ø11	STA $ 2ØØØ
Zero-page	$ 85	2	3	ØØ1	STA $ ØC
Absolut, X	$ 9D	3	5	111	STA $ 12ØØ, X
Zero-page, X	$ 95	2	4	1Ø1	STA $CØ, X
Absolut, Y	$ 99	3	5	11Ø	STA $ 13ØØ, Y
(Indirekt, X)	$ 81	2	6	ØØØ	STA ($ Ø1, X)
(Indirekt), Y	$ 91	2	6	1ØØ	STA ($ Ø2), Y

Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte

7	6	5	4	3	2	1	Ø
1	Ø	1	×	×	×	Ø	Ø

← bit

Mnemonic: STX
Beskrivning: X-registrets innehåll kopieras till en minnesposition.

Flaggor:

N	V		B	D	I	Z	C
		–					

Adressmod	kod	antal bytes	cykler	"X"	exempel
Absolut	$ 8E	3	4	Ø1	STX $ 2ØØØ
Zero-page	$ 86	2	3	ØØ	STX $ CØ
Zero-page, Y	$ 96	2	4	1Ø	STX $ Ø1, Y

Objektkodens uppbyggnad för de olika adressmoderna:

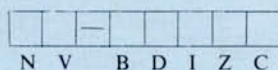
Objektbyte

7	6	5	4	3	2	1	Ø
1	Ø	Ø	×	×	1	1	Ø

← bit

Mnemonic: STY
Beskrivning: Y-registrets innehåll kopieras till en minnesposition.

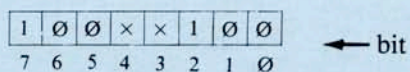
Flaggor:



Adressmod	kod	antal bytes	cykler	"X"	exempel
Absolut	$ 8C	3	4	Ø	$TYS2ØØØ
Zero-page	$ 84	2	3	ØØ	$TYSCØ
Zero-page, X	$ 94	2	4	1Ø	$TYSØ2,X

Objektkodens uppbyggnad för de olika adressmoderna:

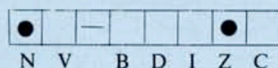
Objektbyte



← bit

Mnemonic: TXA
Beskrivning: Innehållet i X kopieras över till Akumulatorn.

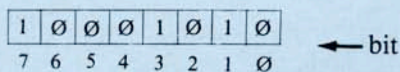
Flaggor:



Adressmod	kod	antal bytes	cykler	"X"	exempel
Implied	$8A	1	2		TXA

Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte



← bit

Mnemonic: TYA
Beskrivning: Innehållet i Y kopieras över till Akumulatorn.

Flaggor:

N	V		B	D	I	Z	C
●		—				●	

Adressmod	kod	antal bytes	cykler	"X"	exempel
Implied	$98	1	2		TYA

Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte

7	6	5	4	3	2	1	Ø
1	Ø	Ø	1	1	Ø	Ø	Ø

← bit

Mnemonic: TAX
Beskrivning: Innehållet i A kopieras över till X-registret.

Flaggor:

N	V		B	D	I	Z	C
●		—				●	

Adressmod	kod	antal bytes	cykler	"X"	exempel
Implied	$AA	1	2		TAX

Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte

7	6	5	4	3	2	1	Ø
1	Ø	1	Ø	1	Ø	1	Ø

Mnemonic: TAY
Beskrivning: Innehållet i A kopieras över till Y-registret.

Flaggor:

N	V		B	D	I	Z	C
●		–				●	

Adressmod	kod	antal bytes	cykler	"X"	exempel
Implied	$A8	1	2		TAY

Objektkodens uppbyggnad för de olika adressmoderna:

Objektbyte

7	6	5	4	3	2	1	Ø
1	Ø	1	Ø	1	Ø	Ø	Ø

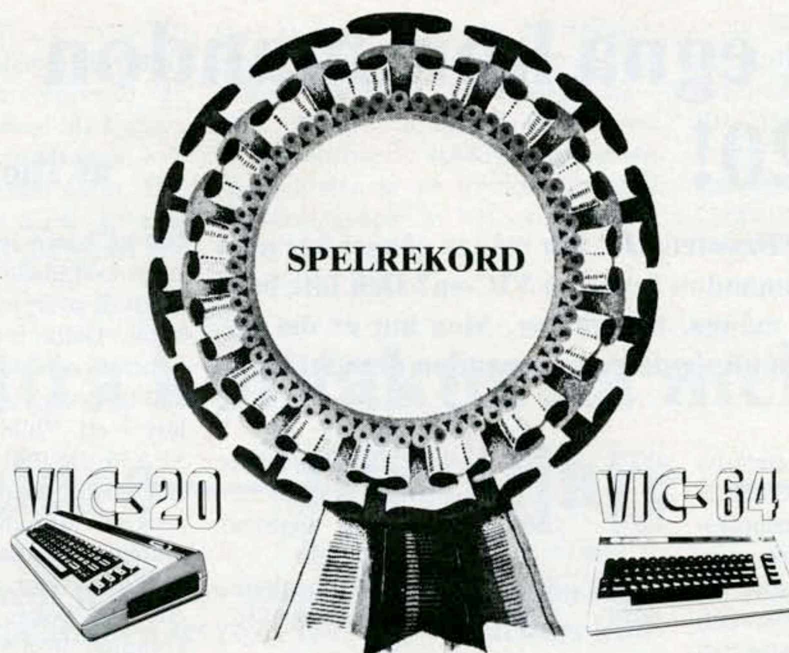
Tabell över ASCII-KOD i BINÄR, OKTAL, DECIMAL och HEXADECIMAL representation.

NUL	0000000	000	000	00	0	0110000	060	048	30
SOH	0000001	001	001	01	1	0110001	061	049	31
STX	0000010	002	002	02	2	0110010	062	050	32
ETX	00000	003	003	03	3	0110011	063	051	33
EOT	0000100	004	004	04	4	0110100	064	052	34
ENQ	0000101	005	005	05	5	0110101	065	053	35
ACK	0000110	006	006	06	6	0110110	066	054	36
BEL	0000111	007	007	07	7	0110111	067	055	37
BS	0001000	010	008	08	8	0111000	070	056	38
HT	0001001	011	009	09	9	0111001	071	057	39
LF	0001010	012	010	OA	:	0111010	072	058	3A
VT	0001011	013	011	OB	;	0111011	073	059	3B
FF	0001100	014	012	OC		0111100	074	060	3C
CR	0001101	015	013	OD	=	0111101	075	061	3D
SO	0001110	016	014	OE		0111110	076	062	3E
SI	0001111	017	015	OF	?	0111111	077	063	3F
DLE	0010000	020	016	10		1000000	100	064	40
DC1	0010001	021	017	11	A	1000001	101	065	41
DC2	0010010	022	018	12	B	1000010	102	066	42
DC3	0010011	023	019	13	C43	1000011	103	067	
DC4	0010100	024	020	14	D	1000100	104	068	44
NAK	0010101	025	021	15	E	1000101	105	069	45
SYN	0010110	026	022	16	F	1000110	106	070	46
ETB	0010111	027	023	17	G	1000111	107	071	47
CAN	0011000	030	024	18	H	1001000	110	072	48
EM	0011001	031	025	19	I	1001001	111	073	49
SUB	0011010	032	026	1A	J	1001010	112	074	4A

ESC	0011011	033	027	1B	K	1001011	113	075	4B
FS	0011100	034	028	1C	L	1001100	114	076	4C
GS	0011101	035	029	1D	M	1001101	115	077	4D
RS	0011110	036	030	1E	N	1001110	116	078	4E
US	0011111	037	031	1F	O	1001111	117	079	4F
SP	0100000	040	032	20	P	1010000	120	080	50
!	0100001	041	033	21	Q	1010001	121	081	51
	0100010	042	034	22	R	1010010	122	082	52
	0100011	043	035	23	T	1010100	124	084	54
%	0100101	045	037	25	U	1010101	125	085	55
&	0100110	046	038	26	V	1010110	126	086	56
'	0100111	047	039	27	W	1010111	127	087	57
(	0101000	050	040	28	X	1011000	130	088	58
)	0101001	051	041	29	Y	1011001	131	089	59
	0101010	052	042	2A	Z	1011010	132	090	5A
+	0101011	053	043	2B	A	1011011	133	091	5B
,	0101101	055	044	2C	Ö	1011100	134	092	5C
—	0101101	055	045	2D	Å	1011101	135	093	5D
.	01011110	056	046	2E		1011110	136	094	5E
/	0101111	057	047	2F	—	1011111	137	095	5F
	F100000	140	096	60					
a	1100001	141	097	61					
b	1100010	142	098	62					
c	1100011	143	099	63					
d	1100100	144	100	64					
e	1100101	145	101	65					
f	1100110	146	102	66					
g	1100111	147	103	67					
h	1101000	150	104	68					
i	1101001	151	105	69					
j	1101010	152	106	6A					
k	1101011	153	107	6B					
l	1101100	154	108	6C					
m	1101101	155	109	6D					
n	1101110	156	110	6E					
o	1101111	157	111	6F					
p	1110000	160	112	70					
q	1110001	161	113	71					
r	1110010	162	114	72					
s	1110011	163	115	73					
t	1110100	164	116	74					
u	1110101	165	117	75					
v	1110110	166	118	76					
w	1110111	167	119	77					
x	1111000	170	120	78					
y	1111001	171	121	79					
z	1111010	172	122	7A					
ä	1111011	173	123	7B					
ö	1111100	174	124	7C					
å	1111101	175	125	7D					
	1111110	176	126	7E					
DEL	1111111	177	127	7F					

Förklaring av kontrolltecken i ASCII-kod

NUL	Null, or all zeros
SOH	Start of heading
STX	Start of text
ETX	End of text
EOT	End of transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horisontal tab
LF	Line feed
VT	Vertikal tab
FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
DLE	Data link escape
DC1	Device control 1
DC2	Device control 2
DC3	Device control 3
DC4	Device control 4
NAK	Negative acknowledge
SYN	Synchronus idle
ETB	End of transmission block
CAN	Cancel
EM	End of medium
SUB	Substitute
ESC	Escape
FS	File separator
GS	Group separator
RS	Record separator
US	Unit separator
SP	Space
DEL	Delete



JELLY MONSTER VIC-1905

1.	1 631 860	Pelle Gustavsson, SUNDBYBERG
2.	763 650	Peter Johansson, LYCKEBY
3.	705 980	Anders Hellman, NOL

RAT RACE VIC-1909

1.	101 520	Pär-Olof Håkansson, BJÄRRED
2.	89 120	Martin Hedberg, SILJANSNÄS
3.	87 000	Niclas Andersson, ESKILSTUNA

SUPERLANDER VIC-1907

1.	88 000	Fredrik Gustafzon, GNESTA
2.	86 000	Thomas Johansson, YSTAD
3.	77 700	Tomas Lundin, SKÖVDE

OMEGA RACE VIC-1924

1.	125 600	Jonas Ragnarsson, BORRBY
2.	105 100	Anders Ragnarsson, BORRBY
3.	23 000	Johan Kuuse, GRÅBO

JUPITER LANDER VIC-1907

1.	139 000	Björn Lindman, HUDDINGE
2.	108 200	Dan Andersson, SÖDRA SANDBY
3.	74 900	Johan Harrysson, VRETA KLOSTER

PANIC 64 (Interceptor)

1.	5 300	Torbjörn Alme'n, SJÖBO
2.	4 200	Charlie Johansson, HANDEN
3.	2 100	Dr Darnio

GRIDRUNNER (LLamasoft)

(Level 1-)

1.	351 820	Thomas Hartman, LULEÅ
2.	260 120	Jöran Omark, ÄLVSBYN
3.	94 180	Niklas Lindberg, VÄSTERVIK

JUMPMAN (EPYX)

(Grand loop)

1.	46 740	Ola Hjalmarsson, UTANSJÖ
2.	37 175	Torbjörn Ragnesjö, UPPSALA

JUMPMAN JR (EPYX)

(Valfri hastighet)

1.	28 000	J. Aspengren, ATRIUM
2.	14 600	P. Andersson, VALÅS
3.	7 425	Lars Karlsson, MÖLNDAL

DIG DUG (Atarisoft)

1.	205 580	STUFFE
2.	74 300	Joakim Lundberg, LINDOME

SPELREKORD

Vi skall försöka att hålla oss till kassettbaserade program för att de flesta har ju endast bandspelare.

Var snälla och skicka in 'rutan' och inte en massa spel med rekord på ett A4 med grannar, mormor, lilla syster, m m.

Kopiera gärna eller skriv av 'rutan' och skicka in fler små i stället.

SPELREDAKTÖREN
C.E.J

☐ VIC 20 ☐ VIC 64 Datum

Namn ....

Adress ....

Postnr, Postadress ....

Spel ....

Erhållna poäng ....

Intygas av målsman eller annan myndig person

Tillverka egna kommandon till VIC-20!

av Thomas Wernersson

Sound, Play, Hires, Plot, Draw etc. Ja, hur många gånger har man inte önskat att dessa kommandon fanns på VIC-en? Och inte bara dessa kommandon, utan många, många fler. Men hur är det nu, kan man på något sätt fixa till sig dessa kommandon utan att lägga ut några pengar?

Svar JA. Detta går att ordna (relativt enkelt). För att kunna göra det måste man ha lite färdighet i maskinspråksprogrammering. Hur man programmerar in sina kommandon bygger på ungefär samma princip som interruptprogrammering, som vi har beskrivit på annan plats i VIC-rapport.

Med normal BASIC-exekvering går det till så här:

1 Datorn exekverar RUN
2 Hämta adress till BASIC-tolk
3 Hoppa till BASIC-tolken
4 Ta in BASIC-kommando (eller karaktär/tecken typ A—0)
5 Utför kommandot
6 Hoppa till "2".

FIGUR 1

Adressen till BASIC-tolken ligger i en vektor med namnet "Start New BASIC Code Link". Denna vektor är placerad i adresserna 776,777. De innehåller i vanliga fall värdena 199 och 228, vilka får datorn att hoppa till hex $C7E4. Med start på $C7E4 ligger den vanliga BASIC-tolken. Den ligger i ROM, så den kan man ju inte gå in i och ändra.

Vad vi behöver göra är att ändra adressen till BASIC-tolken i vektorn så, att den "pekar" till en ny BASIC-tolk någonstans i RAM-minnet. När nu datorn kommer till punkt 2 i figur 1 så laddar den då in adressen som pekar på starten till din BASIC-tolk. På punkt 3 hoppar den dit.

När datorn kommer till din BASIC-tolk så ska den först läsa in så många karaktärer/tecken, som ditt kommando innehåller (alltså bara kommandot, inte ev. parametrar). Inläsningen av tecken sker genom ett subrutinhopp till hex $0073, där en s k Charget-rutin ligger. Denna ligger i Zero-Page och ser ut enligt följande:

```
0073   INC $7A      Ökad lågbyte
                    (Basic Pekare)
0075   BND $0079 Även högbyte?
                    Nej, >0079
0077   INC $7B   Ja, öka högbyte
0079   LDA $1001 Ladda BASIC-
                    tecken
007C   CMP #$3A  Jämför med
                    kolon (:)
007E   BCS $008A  A> = kolon,
                    hoppa till RTS
0080   CMP #$20 Mellanslag?
0082   BEQ $0073 Ja, hämta nästa
tecken (skippa mellanslag)
0084   SEC          Följande fyra in-
0085   SBC #$30  struktioner sätter
0087   SEC          flaggor enligt
0088   SBC #$D0  nedan.
008A   RTS
```

FIGUR 2 Charget-rutin

När denna rutin anropats kommer flaggorna att ha följande betydelse: Carry-flaggan är "clear" (C=0) om A innehåller en siffra, annars är den "set" (C=1). Zero-flaggan är "set" (Z=1) om A innehåller ett kolon eller en nolla, annars är den "clear" (Z=0).

Ett hopp till denna rutin gör, att BASIC-tecken placeras i ackumulatorn. Sedan skall tecknet jämföras med ditt kommando. Om tecknet överhuvud taget hör till ditt kommando, då får du räkna ner BASIC-pekaren (se FIGUR 3) till adressen den hade innan du hoppade till den och sedan hoppa till den vanliga BASIC-tolken. Eftersom det du tittat på kanske är något annat kommando eller tecken som därför måste tolkas av datorn.

```
(i ram)
DEC$7A
BNE$02
DEC$7B
RTS
```

FIGUR 3 Nedräkning

När nu kommandot tolkats så skall du ta in eventuella parametrar och kolla så att de överensstämmer till värde och antal. Detta jobb kallas för Syntaxkontroll och till din hjälp har du ett 200-tal olika små subrutiner som samlats i ett "bibliotek" från $C000—$EA36. De flesta går att använda ganska enkelt, men vissa kan vara mer invecklade att hantera. Jag kan inte presentera dem alla p g a att det skulle ta åtskilliga sidor. Det finns dock i den engelska boken "VIC Revealed" en komplett lista över dem och en beskrivning av de 36 viktigaste på sidorna 71—88. I den kompletta listningen på ett PAUSE-kommando i slutet av denna artikel används inga av dessa s k KERNAL-rutiner då jag vill hålla texten så lätt att förstå som möjligt — även för icke maskinspråksproffs.

Efter att ha passerat Syntax-kontrollen, skall din funktion exekveras. Parametrar bör ha lagrats på Zero-Page för att programmet skall gå fortare och spara minne både i program- och variabelsammanhang.

Under det att exekveringen av ditt kommando pågår så görs hela tiden interrupt, avbrott, därför skall du aldrig koppla ur interrupt med "Interrupt disable". Då kommer bl a klockan att stanna upp under tiden som din funktion "körs". Efter det att funktionen utförts så hoppar du till $67E4, vilket gör att datorn hoppar till den vanliga tolken och att det exekverande BASIC-programmet går vidare.

För er, som läst min andra artikel om interrupt-programmering, är detta med vektor-programmering inte så främmande, eftersom det liknar interrupt ganska väl. Skillnaden är bara att interrupt orsakas regelbundet (normalt 60 ggr/sek, alltså i takt med klockan) medan datorn endast hoppar till din rutin om den stöter på ett BASIC-tecken.

Lägg märke till att ditt kommando lagras byte för byte i ASCII i BASIC-program, medan de "riktiga" kommandona lagras som en enda byte. Den sägs få ett "token"-värde (to-

ken = pant på engelska).

VIC-20s och förresten alla CBM-datorers hårdvara (maskinvara) är mycket flexibel och genom att ändra i dess operativ-system kan du få en helt annorlunda maskin av din dator. Det finns inget som hindrar dig att göra en total ändring av din VIC-20 för att den skall kunna passa till just ditt användningsområde. Du kanske vill ha en 80-teckens bildskärm, ändra I/0 till standard RS-232 m m. Det ena som behövs är en ändring av KERNAL-systemet. Flexibiliteten är en mycket värdefull funktion, skapad av VICs konstruktörer.

Nu hoppas jag bara att jag förklarat detta tillräckligt väl att också du kan använda denna funktion hos din VIC-20. Se följande kompletta listning av ett ''PAUSE''-kommando med förklaringar.

Lycka till!

Tomas Wernersson

Förbättra kontakterna mellan användare — försäljare — hackers

Ett starkt argument för val av dator, är att säljarna efter köpet kan erbjuda kunden service. Här skulle hackers kunna vara till stor nytt. Om säljaren skulle kunna hjälpa hackers till att marknadsföra sig skulle en del av problemen för kunden vara lösta.

Det stora företaget hade infört datoriseringar i sina rutiner. Efter inkörningsperioden så besökte systemeraren på datorföretaget kunden för att kontrollera om allt var som det skulle. Det var inga som helst problem. På vägen ut så råkade han titta in i ett kopieringsrum. Där stod det en flicka och kopierade datalistor för brinnande livet. Systemeraren frågade varför? — ''Vi får bara en lista, men vi behöver fyra'' svarade hon. Därför måste vi kopiera den. Visserligen hade de sagt till, men fått svaret av cheferna att det inte var något att göra åt det. Datorkörningen var tillräckligt dyr.

Att fyrdubbla den bara för att det behövdes fyra listor. Det var ej att tänka på. Systemeraren ändrade lite i utskriftsprogrammet så det blev fyra listor utskrivna. Kostnaden för detta var bara en bråkdel av vad det kostade att kopiera listor.

Så är tyvärr hela datorbranschen i dag.

Om en egen företagare irrar in i en datoraffär, så berättar säljaren hur många tusen bitar det finns i datorn. Den stackars kunden bleknar vid tanken på hur mycket servicen kostar. Och om en enda bit går sönder. Kan han byta ut den själv då.

Skärpning försäljare! Fråga kunden hur mycket denne betalar för bokföringen per år. Ett troligt svar är 20 000:— kronor. Berätta hur mycket ett bokföringsprogram på VIC-64 eller PET kostar. Sälj det sedan. Uppmana företagaren att återkomma efter köpet och berätta vad denne INTE är nöjd med. När kunden återkommer så kan du göra på tre sätt.

1. Rycka på axlarna och beklaga.
2. Hänvisa till det stora programmeringsföretaget, så de får ändra i det befintliga programmet. Eller skriva ett nytt. Att de tar 500:— i timman, det är deras sak.
3. Hänvisa till en hacker.

Här kan en hacker göra stor nytta. Det kan gälla att förändra en utskriftsrutin, poka in företagets firmamärke så det skrivs ut på fakturor och brevpapper. Eller göra ett helt nytt program.

Detta är en helt ny marknad. Kunden kommer knappast att lägga ner några större summor på att ändra programvaran. Alternativet är att låta bli. Så det är ingen risk att ''de stora'' blir trampade på tårna.

Hackern får riktiga uppgifter att arbeta med. Dessutom en slant för besväret. Frågan är bara hur hackerna marknadsför sina tjänster. Här har försäljarna en stor uppgift att fylla. Kan HANDIC och säljarna erbjuda hjälp även efter köpet så är det ett starkt argument vid val av dator. För hackern är det svårt att marknadsföra sig. Det är få företagare som läser datortidningarna. Det bästa är att annonsera i branchtidningar, eller tidningar som vänder sig direkt till egna företagare. Ett tips är att ta kontakt med företagareföreningen där du bor. De kan ge dig tips om vad du skall göra.

Lycka till.

Skäftingebackens datorörnar
Alf Olsson



MASKINSPRÅKSLADDARE FÖR VIC-20

Från vår flerfaldige medarbetare i Kungsbacka, Ingvar Wihlborg, kom följande bidrag, som underlättar tillvaron och minskar lättnaden i plånboken för alla som vill använda maskinspråk i sin programmering på VIC-20.

Maskinkodsladdare för VIC-20

Denna artikel riktar sig till de VIC-rapport-läsare, som följer med i tidningens assemblerskola och som står i begrepp att pröva assemblerprogrammering. Redovisat program underlättar dylik programmering och eliminerar tidsödande skrivning av datsatser och tillhörande "pokeningar".

Programmet gör inga anspråk på att i bland annat flexibilitet etc. kunna mäta sig med plug-in-kassetterna "VIC-MON" och "MIKRO". Använder Du något av nämnda hjälpmedel, måste Du ganska snart expandera din VIC-20 och även anskaffa någon typ av expansionsenhet. Hundralapparna i utlägg ökar då snabbt till tusenlappen. Redovisat program förutsätter dock att din VIC expanderats med 16 K — en vid det här laget synnerligen vanlig företeelse — samt att Du kan offra en timme på inläsning och kassettlagring av programmet. Bortsett från tidsåtgången, är den enda kostnad kassetten. Du kan nu göra dina första övningar i assemblerprogrammering. Tröttnar Du på detta programspråk eller kanske inte har tid eller lust att gå vidare, har Du i varje fall inte satsat stora pengar.

Beskrivna maskinkodsladdare tjänar två syften. Du kan skriva rena maskinspråksprogram, vilka kan "SAVE-as" och "LOADas" samt startas med "RUN" precis som ett Basicprogram. Vidare kan Du göra maskinspråksrutiner, som Du kanske önskar använda som subrutiner i Basicprogram. Vid första alternativet startar alla maskinprogram på adress S120D (dec.4621) och vid alternativ 2 väljer Du själv startadress.

För att Du skall veta var i användarminnet Du arbetar, får Du här en redovisning av minnesuppdelningen.

Adress	4608— 4620	= En s k "Dummy Basic".
Adress	4621—15871	= Area för dina "rena" maskinspråksprogram.
Adress	15872—24319	= Maskinspråksladdarens Basic-program.
Adress	24320—24568	= Plats för maskinspråkssubrutiner.
Adress	24569—24575	= Används för programmet.
Adress	673— 756	= Används av programmet.

Inskrivning av programmet:

Programmet är uppbyggt i två avsnitt. De båda avsnitten benämns "PRE-MON" och "MONITOR". Skriv först in "PRE-MON", spara programmet och exekvera "RUN". Skriv därefter in "MONITOR" och spara detta programavsnitt omedelbart efter "PRE-MON". Du har nu ett acceptabelt hjälpprogram på kassett. Vid användning laddas programmet med "LOAD" på sedvanligt sätt. "RUN" exekveras efter inläsning av "PRE-MON". Ett nytt "LOAD" läser in "MONITOR.

För att underlätta tydandet av listningens grafiska symboler för cursorförflyttningar etc. får Du här en sammanställd förklaring av vissa textraders "otydligheter".

Körning av programmet:

För att programmet skall koda rätt MNEMC matar Du alltid in 6502:ans MNEMCS som fyrställiga kodgrupper (ex.:LDAS) där fjärde tecknet i gruppen utgör kod för adresseringsmode. Följande regler gäller i programmet:
Operander inskrivs hexadecimalt. Bokstäver och siffror, som används i t ex textsträngar skrivs in som A B 1 2 etc. För att underlätta vid just inskrivning av texter finns "ny rad" (CR) på tecknet "pil till vänster" samt mellanslag (space) på tecknet "pil upp". Kanelbullen (snabel a) avslutar programmeringen. Inskrivning av MEM som MNEMC innebär förflyttning till annat minnesutrymme. MEM kan också användas som första instruktion, när Du vill göra programstart på annan adress än $120D.

Rad nr 80 = RVS OM BLACK 8 CURSOR LEFT 12 MELLANSLAG 13 CURS. LEFT.
Rad nr 101 = RVS ON BLACK RVS OFF BLUE.
Rad nr 103 = 2 CURSOR LEFT 2 MELLANSLAG 1 CURSOR UP 1 CURSOR LEFT.
12 MELLANSLAG 14 CURSOR LEFT 1 CURSOR UP.
Rad nr 112 = RVS ON BLACK RVS OFF BLUE.
Rad nr 403 = 4 CURSOR LEFT 3 MELLANSLAG 1 CURSOR UP.
Rad nr 550 = 8 MELLANSLAG 5 CURSOR LEFT.
Rad nr 640 = RVS ON BLACK RVS OFF BLUE.
(Mellanslag = blanktecken eller "spaces").

Efter programstart med "RUN" visas följande text på skärmen: "WAIT FOR LOADING". Maskinspråksrutinerna i programmet "POKEas"in. Efter någon sekund visar skärmen följande utseende:

$ADDR.	OP/& MNEMCS	HEX DEC CODE
120D		

Markören blinkar nu under 'M' i MNEMCS och Du kan börja programmera. I kolumnerna längst till höger skrivs hex. resp. dec. koderna för mnemcs och operander ut i reverserad skrift. Nästa adress visas automatiskt på nästa rad. Upptäcker Du felinmatning på rad du redan exekverat, skriver Du in kanelbullen. Närmast föregående rad suddas ut. Du kan på detta sätt fortsätta raderingen uppåt. Skriver du mnemc som programmet inte förstår, skrivs ett frågetecken ut och Du kan skriva om värdet.

Efter programmet färdigställts, skrivs en asterisk in och Du får upp en meny. Du kan nu disassemblera ditt program. Disassemblering kan stoppas med "RUN/STOP" och fortsätter då Du så önskar med kommandot "CONT". Vill Du stoppa för att göra korrigering, tryck på "S"-tangenten.

RELATIVE & IMPLIED	skrivs med	.	(ex.:TAX.)
ACCUMULATOR	skrivs med	A	(ex.:ASLA)
IMMEDIATE	skrivs med	#	(ex.:LDA#)
ZERO PAGE	skrivs med	Z	(ex.:STAZ)
ZERO PAGE,X e Y	skrivs med	+	(ex.:STA+)
ABSOLUTE	skrivs med	$	(ex.:LDA$)
ABSOLUTE,X	skrivs med	X	(ex.:STAX)
ABSOLUTE,Y	skrivs med	Y	(ex.:LDAY)
INDIRECT,X	skrivs med	1	(ex.:LDA1)
INDIRECT,Y	skrivs med	2	(ex.:STA2)
JUMP INDIRECT	skrivs med	I	(ex.:JMPI)

Ett S-tecken visas och Du skriver nu in hexadressen för raden som skall ändras. En asterisk avslutar alla korrigeringar och Du kan på nytt disassemblera. Vid inskrivning av "MEM" visas också ett S-tecken i högerkant. På samma sätt som vid korrigering skriver Du in en ny adress och programmerar (assemblerar) som vanligt.

Har Du valt "SAVE ON TAPE" ur menyn, uppmanas Du ange "end adress +1". Vid inskrivning av denna adress blankas skärmen. Spara nu ditt maskinprogram genom att skriva "SAVE "PROGRAMNAMN". Vill Du återgå till att använda maskinprogramladdaren skriv SYS 697 och RUN. Ditt sparade maskinprogram laddas med LOAD och startas med RUN.

Som jag inledningsvis nämnde, konkurrerar detta program inte med plug-in-kassetterna, men är en "billig" variant som är värd att pröva. Lycka till och häng med i "assemblerskolan".

Ingvar Wihlborg

Finner Du det jobbigt att skriva in alla data-satser etc., finns programmet att köpa. Priset är 65 kronor, inklusive kassett och porto.

Programmet säljs av:
INGVAR WIHLBORG
(Tfn: 0300-119 80)
PG 47 93 77 6-8

Reset knapp på maskinkods monitorn

Hallå, alla ni assembler programmerare som har tröttnat på ett er VIC-20 "dyker" när ni exekverar era maskinkods program. Har du en HESMON maskinkods monitor så ska det snart vara ordnat. Har du en annan monitor så ta och skruva isär den och kolla om det finns möjlighet att göra det som jag beskriver här nedan.

Det hela går ut på att vi monterar en RESET-knapp i plug-in monitorn. Om då datorn har dykt så är det bara att trycka på denna knapp. Då kommer arbetsbilden fram igen och om du disassemblerar märker du att *programmet finns kvar!* Det du behöver för att få denna suveräna funktion på din VIC-20, är en liten tryckkontakt och två små sladdar, 12—13 cm långa. Skruva nu isär monitorn och plocka ut det lilla gröna kretskortet. Sedan skruvar du ihop "plug-in lådan" igen och borrar ett lagom stort hål i högra "gaveln", för tryckkontakten. Ett alternativ till borrning är att ta en rundfil och fila i de båda lådhalvorna.

Sedan tar du det lilla gröna kretskortet och vänder det med IC:kretsen neråt. Löd nu försiktigt fast de två sladdarna på kretskortskontakt 1 och 3 från *höger* räknat (se fig). OBS det är viktigt att lödningarna inte blir klumpiga eller att de flyter ut på kretskortskontakten.

De andra ändarna av sladdarna löder du fast i tryckkontakten som du sedan skruvar fast i egna lådhalvan, men inte hårdare än att det går att sätta ihop monitorn igen.

Sätt sedan kretskortet på sin rätta plats och skruva ihop maskinkods monitorn.

Skruva sedan åt tryckkontakten ytterligare med en skiftnyckel.

Sedan är det bara att sätta igång med att programmera, utan att vara orolig om datorn dyker.

Morgan Gunnarsson



GRATISANNONSER

VIC-64 5 300:— + PORTO
Dator, bandsp, joystick. Böcker: Basic + avan. prg. på VIC-64, Program. handb. 1, 2, 3. Program: Comal Kalender Sim. Basic spel: Frogger, Othello, Fort Apocalypse mfl, 30 datatidningar (Nypris 7 000:—) GARANTI t o m sept -84. Tel. 011-14 41 12.

BANDSPELARE
till VIC köpes billigt i Göteborg. Tel. 031-29 01 32 (Patrik).

VIC-64 PROGRAM
Spel-, äventyrs-, nytto-program bytes, säljes. Tel. 031/29 01 32 (Patrik).

VIC-64
Spel bytes ev. säljes ring 031-49 37 60, Fredrik.

TILLBEHÖR VIC-20
Expansionsenhet VIC 1020 med 6 portar. Lite använd 1 400:—. Super expander med manual 400:— OBS! säljes. Victor Nilsson, Järfälla 08-760 65 32.

VIC 20 SÄLJES
VIC 20 + bandspelare + joystick + 2 st plugginspel + div. spel, nypris ca 3 500 kr nu 2 500 kr. Tel. efter kl 18.00, 0303-29 148.

VIC-1907 Superlander 100:—, 0531-105 40, Johan.

**** VIC 1515 PRINTER ****
Fullgott skick. 1 250:— (Har kostat 3 250:—). Tel. 031-29 66 18 Göteborg.

VIC-64 SPEL SÄLJES
Gridrunner (cartridge) 150:— 3D time trek (kass) 80:—. Tel. 019-911 14.

SÄLJES EV. BYTES
VIC-20 plug-in, Jelly Monster och Super lander. Skriv till: Dean Tosic, Lokv. 37, 260 33 Påarp.

SÄLJES TILL 64:AN
2 st k. spel: Scramble 64 och Hovver Bovver. Båda för 90 kr st. Ring mellan 18—20. Tel. 031-31 47 65 fråga efter Jan.

VIC-TIPS
Ge eller få tips om de många möjligheterna Du har med VIC 20 eller VIC 64. För amatörer och proffs. Helt gratis. Bifoga svarsporto R. Hall Grönegatan 34, 211 27 Malmö.

VIC 64 PROGRAM
Skicka namn och adress så skickar jag information om mina program. OBS. Du måste ha ''Simon's Basic'' för de flesta programmen. Johan Harrysson, Båtvägen 5, 590 61 Vreta Koster

****V-65, STYRKTIPS OCH LOTTO****
Bra BASIC-program för rankning av travhästar. Bygger på flera års statistik och erfarenhet 150:—, Stryktips med många avancerade funktioner 90:— Lotto 40:— Alla tillsammans 220:—. Tel. 018-11 28 75.

SPEL TILL VIC-20
Spel säljes/bytes, 5—50 kr/st. Ring 019-24 56 40.

64-SPEL SÄLJES OCH BYTES
Space Action bytes eller säljes för 200:— Superscramble original säljes för 75:—. Tel. 08-67 58 78, Christian.





Kommunicera med VIC 64S

Det moderna sättet att få information

Kopplar du ihop ett universalmodem och ett teledataprogram med din VIC 64S så kan du kommunicera med videotexdatabaser via din telefonledning (75/1200 baud). Väljer du ett terminalprogram, kan du med samma modem "tala" med andra VIC 64 ägare eller koppla upp dig mot databassystem som kommunicerar med 300/300 baud (1 baud ≈ 1/10 tecken per sekund).

Med universalmodement kan du alltså välja vilken typ av kommunikation du vill ha. Med videotexprogrammet (Teledata 64) inkopplat kan du t ex koppla upp Datavision, Videodata, Prestel, Telebild, Postel, Micronet osv, osv.

I dessa baser kan du finna massor av information. Köpa en skjorta, sälja en bil, beställa campingplats, skicka meddelanden m m, m m...

Några databaser:

Viewdata AB	08-743 06 65
Datavision	018-901 00
Telebild	08-13 58 00
Videotex telematris	08-99 44 30
Wettergrens	031-11 21 34

Kommunikation är en del av vad VIC 64S kan. I foldern "VIC till vardags" får du mer information om vad VIC kan göra för dig i vardagslivet.

handic electronic ab

Box 1063, 436 00 Askim/Göteborg, Tel. 031/28 97 90
– ett företag i Datatronicgruppen –

Jag vill ha foldern "VIC till vardags"

Namn ______________________

Adress ______________________

Postnr/Ort ______________________

Sänd in kupongen till handic electronic, Box 1063, 436 00 Askim.

Handic Press
Box 42054, 126 12 Stockholm

MASSBREV